

EN0AD-1.0

VAX 11/730 CONSOLE ROM

EP-EN0AD-DL-1.0

1 OF 2 DEC 1987

COPYRIGHT© 1981-86

digital

MADE IN USA

EN0AD-1.0

VAX 11/730 CONSOLE ROM

EP-EN0AD-DL-1.0

2 OF 2 DEC 1987

COPYRIGHT© 1981-86

digital

MADE IN USA

PRODUCT ID: ZZ-EN0AD-1.0
PRODUCT TITLE: 11/730 CONSOLE ROM DOCUMENT
DECO/DEPO: 1.0
DATE: 21-SEP-1987
DEPARTMENT: SASE

COPYRIGHT C) 1981,1986

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

PRODUCT ID: ZZ-EN0AD-1.
PRODUCT TITLE: 11/730 CONSOLE ROM DOCUMENT
DECO/DEPO: 1.0
DATE: 21-SEP-1987
DEPARTMENT: SASE

COPYRIGHT (C) 1981,1986

DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS 01754

THIS SOFTWARE IS FURNISHED UNDER A LICENSE FOR USE ONLY ON A SINGLE COMPUTER SYSTEM AND MAY BE COPIED ONLY WITH THE INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE, OR ANY OTHER COPIES THEREOF, MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY OTHER PERSON EXCEPT FOR USE ON SUCH SYSTEM AND TO ONE WHO AGREES TO THESE LICENSE TERMS. TITLE TO AND OWNERSHIP OF THE SOFTWARE SHALL AT ALL TIMES REMAIN IN DEC.

THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT CORPORATION.

DEC ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DEC.

Error Addr	Code	Seq	Source statement
		1	;
		2	;
		3	RAM003 V00
		4	;
		5	;
=0000		6	
		7	UPC14A EQU 00H ;(RO) UPC bit <14>. Asserted high. <MSB>
		8	
=0001		9	MISC2 EQU 01H ;(RO) Misc bit 2 <MSB>
=0001		10	BOOTSW EQU 01H ;(RO) Boot Switch. Asserted low. <LSB>
		11	
=0002		12	DCLO EQU 02H ;(RO) DC LO Flag. Asserted high. <MSB>
=0002		13	HLTFLG EQU 02H ;(RO) Halt Flag. Asserted low. <LSB>
		14	
=0003		15	ALLOWP EQU 03H ;(RO) Enable Control P Halt Flag. Asserted low. <LSB>
=0003		16	SECURE EQU 03H ;(Same as ALLOWP)
		17	
=0004		18	READJ2 EQU 04H ;(RO) Bit 07 equals J2. <MSB>
=0004		19	APTFLG EQU 04H ;(RO) APT Present Flag. <LSB>
		20	
=0005		21	OK5V EQU 05H ;(RO) 5 volts OK Flag. Asserted high. <MSB>
		22	
=0006		23	SLWCLK EQU 06H ;(RO) Slow (100Hz) Clock. <MSB>
=0006		24	REMOTE EQU 06H ;(RO) Remote Disable Flag. Asserted low. <LSB>
		25	
=0007		26	OKV15 EQU 07H ;(RO) 15 Volts OK Flag. Asserted high. <MSB>
=0007		27	BOOTEN EQU 07H ;(RO) Boot Enable Flag. Asserted low. <LSB>
		28	
=0008		29	WCSB0 EQU 08H ;(WO) WCS Write Data Register, Byte 0
=0009		30	WCSB1 EQU 09H ;(WO) WCS Write Data Register, Byte 1
=000A		31	WCSB2 EQU 0AH ;(WO) WCS Write Data Register, Byte 2
		32	
=001C		33	ITDB EQU 01CH ;(R/W) Timer Data Register
=001D		34	ITCSR EQU 01DH ;(R/W) Timer Status Register
		35	
=0020		36	SUMREG EQU 020H ;(RO) Ready and Done Bit Summary register
		37	;
		38	; 07 06 05 04 03 02 01 00
		39	;
		40	;Timer Power Remote TU-58 Local TU-58 Remote Local
		41	;Int. Fail RCVR RCVR RCVR XMIT XMIT XMIT
		42	; Time Done Done Done Ready Ready Ready
		43	; Out
		44	
=0020		45	CLRCLK EQU 020H ;(WO) Clear CPU Clock Run
=0021		46	SETCLK EQU 021H ;(WO) Set CPU Clock Run
		47	
=0022		48	CLRSS EQU 022H ;(WO) Clear Clock Single Step
=0023		49	SETSS EQU 023H ;(WO) Set Clock Single Step
		50	

Error	Addr	Code	Seq	Source statement
	=0024		51	CLRCSK EQU 024H ;(WO) Clear CSR clock
	=0025		52	SETCSK EQU 025H ;(WO) Set CSR clock
			53	
	=0026		54	CLRUPK EQU 026H ;(WO) Clear the UPC clock
	=0027		55	SETUPK EQU 027H ;(WO) Set the UPC clock
			56	
	=0028		57	SETMCI EQU 028H ;(WO) Set Memory Control Init
	=0029		58	CLRMCI EQU 029H ;(WO) Clear Memory Control Init
			59	
	=002A		60	SETMCK EQU 02AH ;(WO) Set Memory Control Clock
	=002B		61	CLRMCK EQU 02BH ;(WO) Clear Memory Control Clock
			62	
	=002C		63	CLRTMR EQU 02CH ;(WO) Stop timer
	=002D		64	SETTMR EQU 02DH ;(WO) Start timer counting
			65	
	=002E		66	CLRBSY EQU 02EH ;(WO) Clear Unibus Bus Busy
	=002F		67	SETBSY EQU 02FH ;(WO) Set Unibus Bus Busy
			68	
	=0030		69	SETDCL EQU 030H ;(WO) Set DC LO
	=0031		70	CLRDCL EQU 031H ;(WO) Clear DC LO
			71	
	=0032		72	SETACL EQU 032H ;(WO) Set AC LO
	=0033		73	CLRACL EQU 033H ;(WO) Clear AC LO
			74	
	=0034		75	INITL EQU 034H ;(WO) Clear Unibus Init
	=0035		76	INITH EQU 035H ;(WO) Set Unibus Init
			77	
	=0036		78	CLRWCK EQU 036H ;(WO) Clear WCS Write Clock
	=0037		79	SETWCK EQU 037H ;(WO) Set WCS Write Clock
			80	
	=0038		81	CLRCR EQU 038H ;(WO) Clear the CSR Serial Input
	=0039		82	SETCSR EQU 039H ;(WO) Set the CSR Serial Input
			83	
	=003A		84	CLRBLK EQU 3AH ;(WO) Unblock the 100Hz clock
	=003B		85	SFTBLK EQU 3BH ;(WO) Block the 100Hz clock
			86	
	=003C		87	CLRLIT EQU 03CH ;(WO) Clear the Run Light
	=003D		88	SETLIT EQU 03DH ;(WO) Set the Run Light
			89	
	=003E		90	SETPAG EQU 03EH ;(WO) Set the WCS Page
	=003F		91	CLRPAG EQU 03FH ;(WO) Clear the WCS Page
			92	
	=0040		93	TRDB EQU 040H ;(R/W) Remote Data Buffer
	=0041		94	TRSR EQU 041H ;(RO) Remote Status Register
	=0042		95	TRMODE EQU 042H ;(R/W) Remote Mode Registers 1 and 2. See
			96	;comment below
	=0043		97	TRCR EQU 043H ;(R/W) Remote Command Register
			98	
	=0044		99	TUDB EQU 044H ;(R/W) TU-58 Data Buffer
	=0045		100	TUSR EQU 045H ;(RO) TU-58 Status Register

Error Addr	Code	Seq	Source statement
=0046		101	TUMODE EQU 046H ;(R/W) TU-58 Mode Registers 1 and 2. First access
		102	;gets MR1, second gets MR2. Reading the TUCR
		103	;forces the internal pointer back to MR1.
=0047		104	TUCR EQU 047H ;(R/W) TU-58 Command Register
		105	
		106	
		107	
		108	
=0048		109	TTDB EQU 048H ;(R/W) Terminal Data Buffer
=0049		110	TTSR EQU 049H ;(RO) Terminal Status Register
=004A		111	TTMODE EQU 04AH ;(R/W) Terminal Mode Register 1 and 2. First access
		112	;gets MR1, second gets MR2. Reading the CR forces
		113	;the internal pointer back to MR1
=004B		114	TTCR EQU 04BH ;(R/W) Terminal Command Register
		115	
		116	
		117	
		118	
		119	
=0080		120	READ EQU 080H ;(RO) Console Read Register
		121	
=0082		122	CPUACK EQU 082H ;(RO) Acknowledge from CPU. <LSB>
		123	
=0083		124	CPATTN EQU 083H ;(RO) Attention from CPU. <LSB>
		125	
=0084		126	UPC14 EQU 084H ;(RO) UPC bit <14>. <LSB>
		127	
=0085		128	CSR07 EQU 085H ;(RO) CSR bit <07>. <LSB>
		129	
=0086		130	CSR15 EQU 086H ;(RO) CSR bit <15>. <LSB>
		131	
=0087		132	CSR23 EQU 087H ;(RO) CSR bit <23>. <LSB>
		133	
=00A0		134	ENBPAR EQU 0A0H ;(WO) Enable Stall on Parity Error
=00A1		135	DISPAR EQU 0A1H ;(WO) Disable Stall on Parity Error
		136	
=00A2		137	VSHIFT EQU 0A2H ;(WO) Set the V-BUS to Shift Mode
=00A3		138	VNORML EQU 0A3H ;(WO) Set the V-BUS to Normal Mode
		139	
=00A4		140	SETTIM EQU 0A4H ;(WO) Set the Interval Timer Interrupt
=00A5		141	CLRTIM EQU 0A5H ;(WO) Clear the Interval Timer interrupt
		142	
=00A6		143	ENBMEM EQU 0A6H ;(WO) Enable Memory References
=00A7		144	DISMEM EQU 0A7H ;(WO) Disable Memory References
		145	
=00A8		146	SETACK EQU 0A8H ;(WO) Set Console Acknowledge
=00A9		147	CLRACK EQU 0A9H ;(WO) Clear Console Acknowledge
		148	
		149	;NOTE: The Console Acknowledge signal is also used as the UPC
		150	;Serial Input. When used this way the sense is inverted. To simplify

Error Addr	Code	Seq	Source statement
		151	;matters the next two equates are included.
		152	
=00A8		153	CLRUPC EQU 0A8H ;(WO) Clear the V-BUS Serial Input
=00A9		154	SETUPC EQU 0A9H ;(WO) Set the V-BUS Serial Input
		155	
=00AA		156	SETATN EQU 0AAH ;(WO) Set Console Attention
=00AB		157	CLRATN EQU 0ABH ;(WO) Clear Console Attention
		158	
=00AC		159	SETPFI EQU 0ACH ;(WO) Set Power Fail Interrupt
=00AD		160	CLRPFI EQU 0ADH ;(WO) Clear Power Fail Interrupt
		161	
=00AE		162	SETHLT EQU 0AEH ;(WO) Set the Console Halt bit
=00AF		163	CLRHLT EQU 0AFH ;(WO) Clear the Console Halt bit
		164	
=00CC		165	WRITE EQU 0CCH ;(WO) The Console Write Register
		166	
=00EC		167	YBUSRD EQU 0ECH ;(WO) Causes Y-BUS to be loaded into
		168	; Console Read register
		169	
		170	
=005C		171	BACKSL EQU 05CH
=0002		172	RCVDON EQU 02H
=0080		173	UPCFLG EQU 080H
=0020		174	SPACE EQU 020H
=003A		175	COLON EQU 03AH
=0080		176	DACTIV EQU 080H
=0040		177	AACTIV EQU 040H
=0004		178	DFLAG EQU 04H
=0002		179	AFLAG EQU 02H
=0001		180	SPCFLG EQU 01H
=0042		181	AAFLAG EQU 042H ;A Flag and A Active Flag combined
=000F		182	ASMASK EQU 0FH
=00F0		183	DLMASK EQU 0F0H
=0000		184	EXMCOD EQU 00
=0001		185	DEPCOD EQU 01H
=0001		186	UPCADR EQU 01H
=0080		187	USED EQU 080H
=00DF		188	LCMASK EQU 0DFH
=00FE		189	DELMASK EQU 0FEH
=0080		190	ATFLAG EQU 080H
=0000		191	PSLADR EQU 00
=1000		192	CONSMS EQU 01000H
=0457		193	DELAY1 EQU 01111
=0050		194	MAXCNT EQU 80 ;Maximum number of input charactes for CNSOLE
		195	
		196	;Error Codes follow:
		197	;
=0001		198	CRD_CODE EQU 1 ;
=0002		199	CFERR_CODE EQU 2 ;
=0003		200	SYN_CODE EQU 3 ;

Error	Addr	Code	Seq	Source statement
		=0004	201	MEM_CODE EQU 4 ;
		=0005	202	COMM_CODE EQU 5 ;
		=0006	203	CLOCK_CODE EQU 6 ;
		=0007	204	NACKCMD_CODE EQU 7 ;
		=0008	205	NACKDATA_CODE EQU 8 ;
			206	
			207	
		=000D	208	CR EQU 0DH
		=000A	209	LF EQU 0AH
		=0009	210	TAB EQU 09H
		=007F	211	DELETE EQU 07FH
		=001B	212	ALTMOD EQU 01BH ;
		=0012	213	CNTRLR EQU 012H
		=000F	214	CNTRLO EQU 0FH
		=0003	215	CNTRLC EQU 03H
		=0015	216	CNTRLU EQU 015H
		=0013	217	CNTRLS EQU 013H
		=0011	218	CNTRLQ EQU 011H
		=0010	219	CNTRLP EQU 010H ;Control P
		=007B	220	BASE EQU 07BH ;Base loaded into register H for fast decode
		=0018	221	CNTRLX EQU 018H ;Control X(temporary for breadboard)
		=007F	222	PARMSK EQU 07FH ;Parity Mask
		=000D	223	BREAK EQU 0DH ;Break value for USART
		=0004	224	TUINIT EQU 04H ;
		=0015	225	TUPREP EQU 015H ;Normal USART CR value
		=F9C0	226	DIVSR1 EQU 0F9C0H ;
		=FFD8	227	DIVSR2 EQU 0FFD8H ;
		=FFF0	228	DIVSR3 EQU 0FFF0H ;
			229	
		=0016	230	FASUPC EQU 16H
			231	
			232	
		=0007	233	ALARM1 EQU 07H ;This will set the internal pointer
			234	;of the 9513 to point to the Alarm registers
			235	
		=0017	236	MASTER EQU 017H ;This will set the internal pointer of
			237	;the 9513 to point to the Master Mode register
		=0A0C	238	MSTRCD EQU 0A0CH ;Code for Master Mode Register
			239	
		=0001	240	CG1MOD EQU 01 ;This will set the internal pointer of
			241	;the 9513 to point to Counter Group 1 Mode
			242	;Successive writes to the ITDB will cause the
			243	;pointer to increment through the Mode, Load and
			244	;Hold registers of each group
			245	
		=0019	246	HOLD1 EQU 019H ;Set the internal pointer of the 9513
			247	;to point to the Hold register for Counter 1.
			248	;Successive reads will allow the Hold register
			249	;of each group to be examined.
			250	

Error	Addr	Code	Seq	Source statement	
		=001C	251	HOLD4 EQU	01CH ;Point to the group 4 hold register and group
			252		;increment
			253		
		=00A3	254	SAVE12 EQU	0A3H ;Save counters 1 and 2
		=00B8	255	SAVE45 EQU	0B8H ;Save counters 4 and 5
			256		
		=0043	257	LOAD12 EQU	043H ;Load counters 1 and 2 from the Load registers
		=0058	258	LOAD45 EQU	058H ;Load counters 4 and 5
			259		
		=0023	260	ARM12 EQU	023H ;Arm counters 1 and 2 for counting
		=0038	261	ARM45 EQU	038H ;Arm counters 4 and 5
			262		
		=007B	263	LA1245 EQU	7BH ;Load and Arm counters 1,2,4 and 5
			264		
		=0063	265	LARM12 EQU	063H ;Combines Load and Arm operations
		=0078	266	LARM45 EQU	078H ;Load and arm counters 4 and 5
			267		
		=0064	268	LARM3 EQU	064H ;Load and Arm counter 3
			269		
		=0062	270	LARM2 EQU	062H ;Load and arm Counter 2
			271		
		=0083	272	DARM12 EQU	083H ;Stops counting
			273		
		=00F1	274	STEP1 EQU	0F1H ;Single step counter 1
			275		
		=00F2	276	STEP2 EQU	0F2H ;Single step counter 2
			277		
		=000C	278	CGFOUR EQU	0CH ;Point to Counter Four Load register
			279		
		=000D	280	CGFIVE EQU	0DH ;Point to Counter Five Load register
			281		
		=4080	282	STACK EQU	04080H ;
			283		
		=4C00	284	MMSTRT EQU	04C00H ;
			285		
			286		
			287		
			288		
			289	;.....	
			290	;.....	
			291	;The following are entry points into ROM	
			292		
		=0000	293	BEGIN EQU	0 ;
		=000B	294	XVEC EQU	0BH ;
		=0013	295	MODVEC EQU	13H ;
		=001B	296	PWRVEC EQU	1BH ;
		=0040	297	INIVEC EQU	40H ;
		=0043	298	SCVEC0 EQU	43H ;
		=0046	299	GS_VECTOR EQU	46H ;
		=0049	300	RSVEC EQU	49H ;

Error	Addr	Code	Seq	Source statement		
		=004C	301	PWRVE1 EQU	4CH	;
		=004F	302	FU_VECTOR EQU	4FH	;FixUp Vector
		=0052	303	DUMMY1 EQU	52H	;FFU (For Future Use)
		=0055	304	TPINIT EQU	55H	;
		=0058	305	TPINI1 EQU	58H	;
		=0079	306	TUERR2 EQU	79H	;
		=0081	307	TUERR3 EQU	81H	;
		=0086	308	TUERR EQU	86H	;
		=0095	309	TU_SEND_PACKET EQU	95H	;
		=00DE	310	CHKSUM_WORD EQU	0DEH	;
		=00F9	311	COPY EQU	0F9H	;
		=00FB	312	COPY1 EQU	0FBH	;
		=0104	313	SAVESP EQU	104H	;
		=0108	314	SAVSP1 EQU	108H	;
			315			
		=0398	316	CRD_VECTOR EQU	398H	;
			317			
		=080F	318	TALK_VECTOR EQU	80FH	;Vector in ROM to Talk routine
			319			
			320			
		=4080	321	ORG	04080H	;
			322			
			323	;.....		
			324	;		
			325	; The following address are accessed by the ROM and must remain		
			326	; fixed.		
			327	;		
4080	00		328	RAMVAL: DB	0	;Ram Valid Flag
4081	00		329	SWPOS: DB	0	;Switch Position Flag
4082	00		330	BYTSUM: DB	0	;Storage for byte checksum
4083	0000		331	TOCNT: DW	0	;Word wide Time Out Count work area
4085	00		332	TOCNT1: DB	0	;Byte wide Time Out Count work area
4086	0000		333	SPBUFF: DW	0	;Stack Pointer storage area
4088	0000		334	HLBUFF: DW	0	;Storage for HL
			335			
408A	00		336	COLD: DB	0	;Keep the COLD and CODFLG bytes contiguous
408B	00		337	CODFLG: DB	0	
			338			
408C	00		339	RETRY: DB	0	;
			340			
408D	02		341	SNDPAK: DB	02	;Flag byte for commands
408E	0A		342	MSGCNT: DB	10	;Message byte count
408F	02		343	OPCODE: DB	2	;
4090	00		344	MODIFR: DB	0	;Modifier byte
4091	00		345	UNIT: DB	0	;Unit number
4092	00		346	SWITCH: DB	0	;
4093	0000		347	SEQNUM: DW	0	;Sequence number
4095	0000		348	BYTCNT: DW	0	;
4097	0000		349	BLOCK: DW	0	;Block to start from
			350			

Error	Addr	Code	Seq	Source statement	
4099	0000		351	MODADR: DW	0
			352		
409B	00		353	LODFLG: DB	0
409C	00		354	DIRFLG: DB	0
409D	00		355	INDFLG: DB	0
			356		
409E	0000		357	WRDSUM: DW	0
			358		
40A0	0000		359	SNDADR: DW	0
40A2	00		360	SNDCNT: DB	0
			361		
40A3	=0004		362	LODADR: DS	4
			363		
40A7	00		364	LODCNT: DB	0
			365		
40A8	0000		366	PAKCNT: DW	0
			367		
40AA	=000A		368	ENDPAK: DS	10
			369		
40B4	00		370	TEMP: DB	0
			371		
40B5	00		372	PFFLAG: DB	0
			373		
40B6	00		374	PUBFLG: DB	0
			375		
40B7	0000		376	MSGADR: DW	0
			377		
40B9	0000		378	MYPC: DW	0
			379		
40BB	00		380	OFLAG: DB	0
40BC	00		381	OCOPY: DB	0
			382		
			383		
40BD	00		384	WRKMSK: DB	0
40BE	00		385	LRMASK: DB	0
40BF	00		386	MSKCPY: DB	0
			387		
40C0	00		388	SAVCHR: DB	0
			389		
40C1	0000		390	PRIADR: DW	0
40C3	00		391	PRICNT: DB	0
			392		
40C4	0000		393	XADRES: DW	0
40C6	0000		394	XCOUNT: DW	0
			395		
40C8	0000		396	CNTADR: DW	0
40CA	0000		397	BUFADR: DW	0
			398		
40CC	00		399	RETRYI: DB	0
			400		

ZZ-
.MA
RAM
Err

Error	Addr	Code	Seq	Source statement	
	40CD	00	401	SFLAG: DB 0	;
			402		
	40CE	00	403	DELFLG: DB 0	;
			404		
	40CF	00	405	CCFLAG: DB 0	;
			406		
	40D0	00	407	SILFLG: DB 0	;Silo Active Flag
	40D1	00	408	SIBIAS: DB 0	;Storage for Silo Input Bias
	40D2	00	409	SOBIAS: DB 0	;Storage for Silo Output Bias
			410		
	40D3	00	411	RUNFLG: DB 0	;
			412		
	40D4	00	413	NOTYPE: DB 0	;
			414		
	40D5		415	WCS_LOAD_FLAGS:	
	40D5	00	416	DB 0	;Bit 00 = POWER.CMD done, bit 01 = CODE0n.CMD
			417		;done.
			418		
	40D6	00	419	APTLOD: DB 0	;
			420		
	40D7	00	421	SPEND: DB 0	;
	40D8	00	422	QPEND: DB 0	;
			423		
	40D9	00	424	MIPFLG: DB 0	;Modem In Progress flag
			425		
	40DA	00	426	SAVMSK: DB 0	;
			427		
	40DB	00	428	DISCNT: DB 0	;Disconnect Count
			429		
			430		
		=40F0	431	ORG 40F0H	;
			432		
	40F0	0000	433	MSGBUF: DW 0	;
	40F2	00	434	SNDFLG: DB 0	;
	40F3	00	435	CPFLAG: DB 0	;
	40F4		436	SLOW_CLK_FLG:	
	40F4	00	437	DB 0	;
			438		
			439		
		=40FD	440	ORG 40FDH	;Put these last flags at the very end
	40FD		441	CRD_FLAGS:	
	40FD	00	442	DB 0	;Customer Runnable Diagnostics Flags
	40FE	00	443	CTLDIS: DB 0	;Control C and P checking disable flag
	40FF	00	444	NOCHK: DB 0	;Auto LF disable
			445	;	
			446	;	
			447	;	End of this group of fixed addresses
			448	;	
			449	;	*****
			450		

ZZ-
.MA
RAM
Err

Error	Addr	Code	Seq	Source statement
			451	
			452	
			453	;
			454	;
			455	;
		-4100	456	ORG 04100H ;
			457	
			458	
	.100	C3 420C	459	START: JMP MYLOAD ;Go get the rest of the code
	4103	C3 4288	460	GOSV: JMP SETAPT ;Used by X commands out of the ROM
			461	
			462	;
			463	;
			464	;
			465	Start of Parameter passing block. Loads are accomplished by
			466	putting the necessary information in this block and then calling
			467	LDXPRP.
			468	UNITX: DB 0 ;
			469	FNBUF: DB 'CONSOL' ;
			470	DOTBUF: DB ' ' ;
			471	EXTBUF: DB 'EXE' ;
			472	SBUFR: DW 04C00H ;
			473	DW 0 ;
			474	QUAL: DB 6 ;
			475	SUCCE: DB 0 ;
			476	
	4117	C3 4500	477	LODVEC: JMP LDXPRP ;Go to this entry point
	411A	C3 46CB	478	DIRVEC: JMP LSTPRP ;Go to this entry point
	411D	C3 4C22	479	PARERR: JMP PARVEC ;Go to this vector
	4120	C3 4C25	480	ACLO: JMP ACVECT ;Jump to the ACLO Vector
			481	
	4123	CD 0108	482	GLVECT: CALL SAVSP1 ;
	4126	C3 4A4D	483	JMP GETLIN ;
			484	
	4129	CD 0108	485	GCVECT: CALL SAVSP1 ;
	412C	E7	486	RST 4 ;Go to GETCHR
	412D	C9	487	RET ;Leave
			488	
	412E	CD 0108	489	SLVECT: CALL SAVSP1 ;Go mark the SP value
	4131	F7	490	RST 6 ;Go to Send Line
	4132	C9	491	RET ;Leave
			492	
	4133	CD 0108	493	SLVEC1: CALL SAVSP1 ;Go mark the SP value
	4136	EF	494	RST 5 ;Special entry to SNDLIN
	4137	C9	495	RET ;Leave
			496	
	4138	CD 0108	497	SCVECT: CALL SAVSP1 ;Go mark where the SP is
	413B	DF	498	RST 3 ;Go to SNDCHR
	413C	C9	499	RET ;Leave

ZZ-ENOAD-1.0		ENOAD.LIS		N 1		6-NOV-1987		Fiche 1 Frame N1		Sequence 13	
.MAIN.		CR8085/11 version 20.41.15		9-Jan-84 12:56:53		Page 1-10					
RAM003.SRC											
Error	Addr	Code	Seq	Source statement							
			500								
	413D	CD 0108	501	RSVECT:	CALL	SAVSP1	;Go mask the SP value				
	4140	CD 0049	502		CALL	RSVEC	;Go read the top of the silo				
	4143	78	503		MOV	A,B	;Put the character in A for ENKAA				
	4144	C9	504		RET		;Leave				
			505								
	4145	CD 0108	506	GSVECT:	CALL	SAVSP1	;Go mark the SP value				
	4148	CD 0046	507		CALL	GS_VECTOR	;Go get the top of the silo				
	414B	78	508		MOV	A,B	;Put the character in A for ENKAA				
	414C	C9	509		RET		;Leave				
			510								
	414D		511	RET_VECTOR:							
	414D		512	T_VECTOR:							
	414D	C3 4B3C	513		JMP	CRD_RD_TEST	;Go check for CRD or RD.				
			514								
	4150	00	515	DEFDRV:	DB	0	;				
			516								
	4151	C3 001B	517	PUVEC:	JMP	PWRVEC	;Go to Power Up				
			518								
	4154	45 4E 4B	519	MICHON:	DB	'ENKAA',0,'.EXE'					
	4157	41 41 00									
	415A	2E 45 58									
	415D	45									
			520	;							
			521	; End of Parameter block							
			522	;							
			523	;*****							
			524								
	415E	0000	525	REMAIN:	DW	0					
	4160	0000	526	QUO:	DW	0					
			527								
	4162	0000	528	CMPADR:	DW	0	;				
			529								
			530								
	4164	00	531	FNDFLG:	DB	0					
			532								
	4165	00	533	LODDST:	DB	0					
			534								
	4166	0000	535	LENGTH:	DW	0					
			536								
	4168	0000	537	TMPBUF:	DW	0	;Temp buffer				
	416A	0000 0000	538	TEMP1:	DW	0,0	;				
	416E	0000 0000	539	BYTES:	DW	0,0	;				
	4172	0000	540	PACKS:	DW	0	;				
			541								
	4174	0000	542	BLKNUM:	DW	0	;				
			543								
	4176	0000	544	POINTR:	DW	0					
			545								
	4178	00	546	ENTYPE:	DB	0					

Error	Addr	Code	Seq	Source statement
			547	
4179	00		548	MYCNT: DB 0 ;Count for booting
			549	
			550	
417A	00		551	BYTE00: DB 0 ;
417B	00		552	BYTE01: DB 0 ;
			553	
417C	0000		554	DIRADR: DW 0 ;
			555	
417E	00		556	SEGNUM: DB 0 ;
			557	
417F	0000		558	INCCNT: DW 0 ;
			559	
4181	00		560	ALTFLG: DB 0 ;
4182	00		561	CMDVAL: DB 0 ;
			562	
4183	01		563	CMPTB1: DB 01 ;
4184	80		564	DB 128 ;
			565	
4185	02		566	CMPTB2: DB 02 ;
4186	0A		567	DB 10 ;
			568	
4187	00		569	COUNT: DB 0
			570	
4188	04		571	XFRPAK: DB 04
4189	00		572	DB 0
418A	=0004		573	XFRADR: DS 4
418E	=0004		574	XFRCNT: DS 4
			575	
4192	=0002		576	XFRBUF: DS 2
			577	
4194	00		578	DIRVAL: DB 0
4195	00		579	DIRUNI: DB 0
			580	
4196	00		581	CLKFLG: DB 0 ;
			582	
4197	00		583	PARFLG: DB 0
4198	00		584	PARTMP: DB 0 ;
			585	
4199	00		586	SAVCNT: DB 0 ;
			587	
419A	00		588	SAVEB0: DB 0 ;
419B	00		589	SAVEB1: DB 0 ;
419C	00		590	COPYB0: DB 0 ;
419D	00		591	COPYB1: DB 0 ;
419E	0000 0000		592	SAVE: DW 0,0 ;Reserve 4 byte of storage
			593	
41A2	0000		594	PNTR1: DW 0
			595	
41A4	=0003		596	CSRBUF: DS 3

Erro	Addr	Code	Seq	Source statement
41A7	=0003	597	NOPBUF: DS	3
41AA	=0004	598	SWAP: DS	4
41AE	=0002	599	SAVUPC: DS	2
41B0	0000	600	TRKUPC: DW	0
41B2	0000	601	TRKUP1: DW	0
		602		
41B4	00	603	LIMIT: DB	0
41B5	0000	604	LIMSP: DW	0
		605		
41b7	00	606	WARM: DB	0
		607		
41B6	0000	608	SAVADR: DW	0
		609		
41BA	03 5E 55	610	CONTLU: DB	3,'^U',CR
41BD	0D			
		611		
41BE	03 5E 52	612	CONTLR: DB	3,'^R',CR
41C1	0D			
		613		
41C2	0B	614	LSTBUF: DB	11
	=0006	615	DS	6
41C9	2E	616	DB	'.'
	=0003	617	DS	3
41CD	0D	618	DB	CR
		619		
41CE	43 4F 4E	620	CONSOL: DB	'CONSOL.EXE'
41D1	53 4F 4C			
41D4	2E 45 58			
41D7	45			
		621		
41D8	06 20 20	622	DRVFIL: DB	6,' DD0:'
41DB	44 44 30			
41DE	3A			
		623		
41DF	0F 0D 0A	624	CONMSG: DB	15,CR,LF
41E2	20 20 43	625	DB	' CONTINUING'
41E5	4F 4E 54			
41E8	49 4E 55			
41EB	49 4E 47			
41EE	0D	626	DB	CR
		627		
41EF	13	628	FNFMMSG: DB	19
41F0	3F 34 30	629	DB	'?40 FILE NOT FOUND'
41F3	20 20 46			
41F6	49 4C 45			
41F9	20 4E 4F			
41FC	54 20 46			
41FF	4F 55 4E			
4202	44			
		630		

Error	Addr	Code	Seq	Source statement
	4203	03 3E 3E	631	PROMPT: DB 3,'>>>' ;
	4206	3E		
	4207	04 4D 49	632	
	420A	43 3E	633	MICPRO: DB 4,'MIC>' ;
			634	
			635	;.....
			636	;.....
			637	
			638	;.....
			639	
			640	; MYLOAD: Self Load routine
			641	; MYLDER: Self Load Error handler
			642	
			643	; MYLOAD is entered from START, which is where the ROM jumps when it
			644	; manages to load from the Boot Block of the TU-58 tape. Here we attempt to
			645	; find CONSOL.EXE or ENKAA.EXE (the Micro Monitor). If neither can be found on
			646	; either drive, then we go back to the ROM because not enough code has been
			647	; loaded to do anything interesting.
			648	
			649	; MYLDER is used to print messages regarding any of the errors that might
			650	; occur as we search for CONSOL.EXE or ENKAA.EXE on either Drive 1 or 0.
			651	
			652	
			653	; REGISTERS: A = General use
			654	; B = Counter
			655	; DE = Source String pointer
			656	; HL = Destination pointer.
			657	; General purpose pointer.
			658	; Output string (ASCII) pointer.
			659	
			660	; SUBROUTINES: COPY1
			661	; LDXPRP
			662	
			663	; FLAGS AND COUNTERS:
			664	
			665	; (CLEARED) MYCNT
			666	; (SET)
			667	; (READ) MYCNT- Limits number of passes through
			668	; MYLOAD to 4.
			669	; - Used to determine when to flip UNIT and
			670	; CODFLG.
			671	; UNIT- Determines which unit we ask LDXPRP
			672	; to access.
			673	; CODFLG- Determines which file name argument
			674	; to pass to LDXPRP.
			675	; SUCCES
			676	; (WRITTEN) MYCNT- Incremented each time we pass through
			677	; MYLOAD.
			678	; UNITX- Copied from UNIT before calling LDXPRP.

Error Addr Code Seq Source statement

```

679 ;
680 ;
681 ;
682 ;
683 ;
684 ;
685 ;
686 ;
687 ;
420C AF 688 MYLOAD: XRA A
420D 32 4179 689 STA MYCNT
4210 3A 4091 690 1$: LDA UNIT
4213 32 4106 691 STA UNITX
4216 21 4107 692 LXI H,FNBUF
4219 11 41CE 693 LXI D,CONSOL
421C 06 0A 694 MVI B,10
421E 3A 408B 695 LDA CODFLG
4221 1F 696 RAR
4222 D2 4228 697 JNC 2$
4225 11 4154 698 LXI D,MICMON
4228 CD 00FB 699 2$: CALL COPY1
422B CD 4500 700 CALL LDXPRP
422E 3A 4116 701 LDA SUCCES
4231 A7 702 ANA A
4232 CA 4296 703 JZ SETVAL
4235 3E 0D 704 MVI A,CR
4237 DF 705 RST 3
4238 CD 4256 706 CALL MYLDER
423B 21 4179 707 LXI H,MYCNT
423E 34 708 INR M
423F 7E 709 MOV A,M
4240 21 4091 710 LXI H,UNIT
4243 FE 02 711 CPI 2
4245 DA 4253 712 JC 4$
4248 CA 4250 713 JZ 3$
424B FE 03 714 CPI 3
424D CA 4253 715 JZ 4$
4250 21 408B 716 3$: LXI H,CODFLG
4253 7E 717 4$: MOV A,M
4254 EE 01 718 XRI 01
4256 77 719 MOV M,A
4257 3A 4179 720 LDA MYCNT
425A FE 04 721 CPI 4
425C CA 4C2B 722 JZ PVECTR
425F 21 41DF 723 LXI H,CONMSG
4262 F7 724 RST 6
4263 C3 4210 725 JMP 1$
726
4266 2A 40B7 727 MYLDER: LHLD MSGADR
4269 F7 728 RST 6

```

UNIT- Flipped on passes through MYLOAD
 when MYCNT is odd. Keeps track of unit
 we are currently accessing.
 CODFLG- Flipped on passes through MYLOAD when
 MYCNT is even. Keeps track of which
 file we are searching for.

```

;Make a 0
;Zero out this counter
;Get the unit we powered up with
;Put it here
;Point to the File Name area
;Point to this source
;Set up a count of 10 decimal
;Get the code flag
;Check the LSB
;If zero, don't change the source
;Otherwise, change it
;Go copy it over
;Go do the load
;Get the success code
;Check for 0
;Go set the Ram Valid bit
;Get a carriage return
;Go send it
;Go print an error message
;Point to this count
;Bump it
;Get it into A
;Point to the present Unit number
;Check the count
;If it is 1, go flip the Unit Number
;If it is 2, go flip the Code Flag
;See if it is 3
;If it is 3, go flip the Unit Number
;Flip the Code Flag
;Get whatever we are pointing at
;Flip it
;Save the new value
;Check the count
;We're done if it equals 4
;Use this vector to go back to the ROM
;Point the Continuing message
;Go send it out
;Go try again
;Get the address of the error message
;Go print it

```


Error	Addr	Code	Seq	Source statement
426A	3A 4116		729	LDA SUCCES ;Get the success code
426D	FE 02		730	CPI 2 ;Check for Drive (Init) type error
426F	C8		731	RZ ;Leave if it is
4270	3A 4091		732	LDA UNIT ;Get the Unit #
4273	F6 30		733	ORI 030H ;Make it ASCII
4275	32 41DD		734	STA DRVFIL+5 ;Store it
4278	21 41D8		735	LXI H,DRVFIL ;Point to this message
427B	F7		736	RST 6 ;Go send it out
427C	3A 4116		737	LDA SUCCES ;Get the success code byte
427F	3D		738	DCR A ;See if it is one by trying to make it zero
4280	C0		739	RNZ ;Leave if not
4281	21 4107		740	LXI H,FNBUF ;Point to the file name
4284	3E 0A		741	MVI A,10 ;Count for printing
4286	EF		742	RST 5 ;Go print it
4287	C9		743	RET ;Leave
			744	;
			745	;.....
			746	;
			747	;.....
			748	;
			749	; SETAPT
			750	; SETVAL
			751	;
			752	; SETAPT is the entry point used to transfer control from ROM to RAM
			753	;as the result of a successful downline load (X/L 4100 3F00) to load up the
			754	;RAM when a TU-58 is unavailable. We come here from the GOSV vector that the
			755	;ROM X command jumps to when it finishes. We set up a few flags and then fall
			756	;through to SETVAL.
			757	;
			758	; SETVAL sets up the Default Drive number, sets the Ram Valid flag and
			759	;then goes to GOSTRT. GOSTRT is a Vector that's loaded when either CONSOL.EXE
			760	;or ENKAA.EXE is loaded. If CONSOL.EXE is loaded, it will send us to the code
			761	;that tries to load up the Micro Code and Boot up the machine.
			762	;
			763	;
			764	; REGISTERS: A = General use
			765	; HL = Pointer to RAMVAL
			766	;
			767	; FLAGS: (CLEAR) CTLDIS- Control Character Checking Disable
			768	;
			769	; (SET) APTLOD- Loaded from APT
			770	; CODFLG- Code Flag. (0=CONSOL.EXE, 1=ENKAA.EXE)
			771	; DEFDRV- Default Drive (TU-58)
			772	; RAMVAL- Ram Valid (CONSOL.EXE or ENKAA.EXE loaded)
			773	;
4288	AF		774	SETAPT: XRA A ;Make a 0
4289	32 40FE		775	STA CTLDIS ;Turn this off
428C	3C		776	INR A ;Make a 1
428D	32 40D6		777	STA APTLOD ;Set the Loaded via APT flag
4290	3A 4C2E		778	LDA CODFL1 ;Get the assembled in version of this flag

Error Addr	Code	Seq	Source statement
4293	32 408B	779	STA CODFLG ;Put it where it belongs
4296	3A 4106	780	SETVAL: LDA UNITX ;Get the unit number
4299	32 4150	781	STA DEFDRV ;Set the default drive number
429C	21 4080	782	LXI H,RAMVAL ;Point to this flag byte
429F	36 01	783	MVI M,1 ;Set it
42A1	C3 4C00	784	JMP GO_START ;This will vector us to the correct place
		785	;
		786	;*****
		787	;
		788	;
		789	;*****
		790	;
		791	;
		792	;
		793	;
		794	;
		795	;
		796	;
		797	;
		798	;
42A4	3E 01	799	TUERR1: MVI A,1 ;Error code for Success byte
42A6	21 41CF	800	LXI H,FNFMSG ;Address of File Not Found Message
42A9	C3 0086	801	JMP TUERR ;Go to this shared code
		802	;
		803	;*****
		804	;
		805	;
		806	;*****
		807	;
		808	;
		809	;
		810	;
		811	;
		812	;
		813	;
		814	;
		815	;
		816	;
		817	;
		818	;
		819	;
		820	;
42AC	21 41BE	821	CNTRLR: LXI H,CNTRLR ;Point to the ^R string
42AF	F7	822	RST 6 ;Go print it
42B0	21 4203	823	LXI H,PROMPT ;Point to my prompt
42B3	3A 408B	824	LDA CODFLG ;See who is here
42B6	1F	825	RAR ;Is it set?
42B7	D2 42BD	826	JNC CNTRLR1 ;Jump if not set
42BA	21 4207	827	LXI H,MICPRO ;Point to the MIC Prompt
42BD	F7	828	CNTRLR1: RST 6 ;Go print it

Error	Addr	Code	Seq	Source statement
	42BE	2A 40C8	829	LHLD CNTADR ;Point to the character count
	42C1	7E	830	MOV A,M ;Get the count
	42C2	A7	831	ANA A ;See if zero
	42C3	CA 4A58	832	JZ GETLIO ;If zero,don't do anything
	42C6	F7	833	RST 6 ;Go print it
	42C7	AF	834	XRA A ;Make a 0
	42C8	32 40CE	835	STA DELFLG ;Zero the Delete Flag
	42CB	C3 4A58	836	JMP GETLIO ;Go get the next character
			837	;
			838	;.....
			839	;
			840	;
			841	;.....
			842	;
			843	; CNTLU: Control U handler
			844	;
			845	; This handler prints an ^U, clears the Command Line Buffer Count
			846	;and then uses the the Control P vector to get to the correct idle loop in
			847	;either CONSOL.EXE or ENKAA.EXE
			848	;
			849	;
			850	; REGISTERS: A = General use
			851	; HL = Pointer to ^U string
			852	; Pointer to Command Line count
			853	;
			854	; SUBROUTINES: SNDLIN- Send Line
			855	;
			856	;
	42CE	21 41BA	857	CNTLU: LXI H,CNTLU ;Point to this string
	42D1	F7	858	RST 6 ;Go print it
	42D2	AF	859	XRA A ;Make a 0
	42D3	2A 40C8	860	LHLD CNTADR ;Get the address of the count
	42D6	77	861	MOV M,A ;Zero the count
	42D7	C3 4C2B	862	JMP PVECTR ;Go to the correct idle loop
			863	;
			864	;.....
			865	;
			866	;
			867	;
			868	;.....
			869	;
		=42F6	870	ORG 42F6H ;Don't change this address
			871	;
	42F6	0000	872	DIRSEG: DW 0 ;
	42F8	0000	873	NXTSEG: DW 0 ;
	42FA	0000	874	HIGHST: DW 0 ;Number of Highest Segment open
	42FC	0000	875	EXTRAS: DW 0 ;Number of Extra Words per entry
	42FE	0000	876	STRBLK: DW 0 ;Block where the files in this segment begin
			877	;
			878	;.....

Error	Addr	Code	Seq	Source statement
			879	;*****
			880	
		=4300	881	ORG 4300H ;Now we will set aside a 512 byte area for the
			882	;Directory Buffer
			883	
4300		0000	884	DIRBUF: DW 0 ;
4302		=01FE	885	NAMES: DS 510 ;
			886	
			887	
			888	
			889	;*****
			890	;*****
			891	;
		=4500	892	ORG 04500H ;
			893	;
			894	; LDXPRP: This is the TU-58 handler. The parameter passing block
			895	;at 4105H must be loaded with the parameters controlling the Load prior
			896	;to calling LDXPRP. The handler then attempts to locate and load the
			897	;file. To do this it uses the List handler (which performs the Directory
			898	;function) as a subroutine. Hooks have been added to the List handler
			899	;to allow it to interact with LDXPRP, the LODFLG and FNDFLG in particular.
			900	; If the file is found, the Load handler uses the information generated
			901	;by List regarding the location and length of the file on the tape to
			902	;initiate a read from the tape. There may be multiple requests because the
			903	;driver divides the block count by 16 and asks for the file in chunks that
			904	;are no greater than 16 blocks (8k bytes) at a time.
			905	;
			906	; REGISTERS: A =
			907	; B =
			908	; C =
			909	; D =
			910	; E =
			911	; H =
			912	; L =
			913	; BC =
			914	; DE =
			915	; HL =
			916	;
			917	; SUBROUTINES: LIST8
			918	; LIST
			919	; COPY
			920	; SUBDIV
			921	; WRTNOP
			922	; LDUPC
			923	; LOADA
			924	; LDCSR1
			925	;
			926	;
			927	; FLAGS: (CLEAR) FNDFLG- File Found Flag
			928	;

Error	Addr	Code	Seq	Source statement	
			929	:	(SET) LODFLG- Load In Progress flag
			930	:	
			931	:	
4500	CD 0104		932	LDXPRP: CALL	SAVESP ;Go save the stack value
4503	3A 4106		933	LDA	UNITX ;Get the specified Unit #
4506	32 4091		934	STA	UNIT ;Put it here
4509	CD 0058		935	CALL	TPINI1 ;Go init the drives
450C	AF		936	XRA	A ;Make a 0
450D	32 4164		937	STA	FNDFLG ;Zero the File Name Found flag
4510	3C		938	INR	A ;Make a 1
4511	32 409B		939	STA	LODFLG ;Set the load flag
4514	3A 4194		940	LDA	DIRVAL ;Check the directory valid flag
4517	1F		941	RAR	;Is it set
4518	D2 4544		942	JNC	DIFUNI ;Jump if not
451B	3A 4091		943	LDA	UNIT ;Get the unit number
451E	21 4195		944	LXI	H,DIRUNI ;Point to the unit number of the Directory
4521	BE		945	CMP	M ;Are they the same?
4522	C2 4544		946	JNZ	DIFUNI ;Jump if not
4525	3A 417E		947	LDA	SEGNUM ;Get the segment number
4528	FE 01		948	CPI	1 ;Check it
452A	C2 4544		949	JNZ	DIFUNI ;Jump if it is not a 1
			950		;NOTE:Accumulator now contains 1
452D	21 430A		951	LXI	H,DIRBUF+10 ;Set up this pointer
4530	22 417C		952	SHLD	DIRADR ;Save the pointer
4533	21 FFFE		953	LXI	H,-2 ;Set up a minus 2 bias
4536	39		954	DAD	SP ;Add the SP value in
4537	22 41B5		955	SHLD	LIMSP ;Save it
453A	CD 46E3		956	CALL	LIST8 ;Use this entry. (NOTE: A=1 from above, so
			957		;Limit Flag will be set at LIST8 entry)
453D	3A 4164		958	LDA	FNDFLG ;Check the File Found Flag
4540	1F		959	RAR	;Is it set?
4541	DA 454E		960	JC	LOADX3 ;Jump if found
4544	CD 46D9		961	DIFUNI: CALL	LIST ;Go find the file
4547	3A 4164		962	LDA	FNDFLG ;Get the File Found Flag
454A	1F		963	RAR	;Check it
454B	D4 42A4		964	CNC	TUERR1 ;Jump if it wasn't found
454E	21 40A3		965	LOADX3: LXI	H,LODADR ;Point to the Load Address Buffer
4551	11 4111		966	LXI	D,SBUFFR ;Point to the Start Address Buffer area
4554	CD 00F9		967	CALL	COPY ;Go copy the Start Address into the Load Address
4557	2A 4166		968	LHLD	LENGTH ;Get the length
455A	22 415E		969	SHLD	REMAIN ;Put it in the Remainder area
455D	11 FFF0		970	LXI	D,DIVSR3 ;Get the divisor(2's comp of 010H)
4560	CD 4875		971	CALL	SUBDIV ;Go see how many times it goes
4563	3A 4115		972	LDA	QUAL ;Check for load to WCS
4566	FE 07		973	CPI	07 ;Is it?
4568	C2 45A1		974	JNZ	LOADA ;Jump if it isn't
456B	D3 20		975	OUT	CLRCLK ;Stop the clocks
456D	D3 A7		976	OUT	DISHMEM ;Turn off memory
456F	CD 4993		977	CALL	WRTNOP ;Write a NOP to the CSR
4572	2A 40A3		978	LHLD	LODADR ;Get the address

Error Addr	Code	Seq	Source statement	
4575	22 41AA	979	SHLD SWAP	;Store it here
4578	22 41B0	980	SHLD TRKUPC	;Save a copy here also
457B	21 41AA	981	LXI H,SWAP	;Point to it
457E	CD 49D1	982	CALL LDUPC	;Go put it in the UPC
4581	2A 41AA	983	LHLD SWAP	;Get the old UPC
4584	22 41AE	984	SHLD SAVUPC	;Save it for a bit
4587	AF	985	XRA A	;Make a 0
4588	32 4187	986	STA COUNT	;Initialize the B0,B1,B2 counter
458B	CD 45A1	987	CALL LOADA	;Go do the load
458E	21 41AE	988	LXI H,SAVUPC	;Point to where the UPC was saved
4591	CD 49D1	989	CALL LDUPC	;Go restore it
4594	CD 49A7	990	CALL LDCSR1	;Go restore the CS ^R
4597	D3 A6	991	OUT ENBMEM	;Turn on memory
4599	3A 4196	992	LDA CLKFLG	;Check the clock f ag
459C	1F	993	RAR	;Is it set?
459D	D0	994	RNC	;Leave if not
459E	D3 21	995	OUT SETCLK	;Turn if back on if set
45A0	C9	996	RET	;Leave
		997		
45A1	2A 4160	998	LOADA: LHLD QUO	;Get the quotient
45A4	7C	999	MOV A,H	;Put the high byte into A
45A5	B5	1000	ORA L	;Or in the low byte
45A6	CA 45B6	1001	JZ LOADB	;Jump if 0
45A9	2B	1002	DCX H	;Decrement the count
45AA	22 4160	1003	SHLD QUO	;Save the new value
45AD	21 0010	1004	LXI H,016	;This is the blocking size
45B0	CD 45C3	1005	CALL LOAD1	;Go do the load
45B3	C3 45A1	1006	JMP LOADA	;Jump back
		1007		
45B6	2A 415E	1008	LOADB: LHLD REMAIN	;Get the remainder
45B9	7C	1009	MOV A,H	;Must check for 0 Remainder
45BA	B5	1010	ORA L	;Or in the low byte
45BB	C4 45C3	1011	CNZ LOAD1	;Call Load1 one more time if Remainder not 0
45BE	AF	1012	XRA A	;Make a 0
45BF	32 409B	1013	STA LODFLG	;Zero this flag
45C2	C9	1014	RET	;Leave
		1015		
45C3	3A 4187	1016	LOAD1: LDA COUNT	;Get the count value
45C6	32 4199	1017	STA SAVCNT	;and save a copy
45C9	22 4168	1018	SHLD TMPBUF	;Save the block size
45CC	06 09	1019	MVI B,09	;Count for shifting
45CE	CD 46BE	1020	CALL MULSHF	;Go do a shift multiply
45D1	22 4095	1021	SHLD BYTCNT	;The result is the Byte count
45D4	22 418E	1022	SHLD XFRCNT	;Save it here for adding later
		1023		;and for sending to the CPU
45D7	2A 4168	1024	LHLD TMPBUF	;Get the block size again
45DA	06 02	1025	MVI B,02	;Multiply by 4
45DC	CD 46BE	1026	CALL MULSHF	;Go do it
45DF	22 40A8	1027	SHLD PAKCNT	;This is the number of 128 byte packets
45E2	22 4172	1028	SHLD PACKS	;Save this value for possible retries

Error	Addr	Code	Seq	Source statement
	45E5	2A 4174	1029	LHLD BLKNUM ;Get the Starting Block
	45E8	22 4097	1030	SHLD BLOCK ;Put it in the Send Packet
	45EB	3E 09	1031	MVI A,09 ;Retry count
	45ED	32 408C	1032	STA RETRY ;Store it
	45F0	21 418A	1033	LXI H,XFRADR ;Point to a the XFR address area
	45F3	11 40A3	1034	LXI D,LODADR ;Point to the current address
	45F6	CD 00F9	1035	CALL COPY ;Save a copy of it
	45F9	21 4188	1036	LOAD1X: LXI H,XFRPAK ;Point to the XFR Packet just in case
	45FC	3A 4115	1037	LDA QUAL ;Get the Qual byte
	45FF	FE 03	1038	CPI 03 ;Check for P or V
	4601	DC 514E	1039	CC SNDMSG ;Go send the XFR Packet
	4604	AF	1040	XRA A ;Make a 0
	4605	32 4082	1041	STA BYTSUM ;Zero the bytsum
	4608	CD 0095	1042	CALL TU_SEND_PACKET ;Go send the command packet
	460B	3A 4115	1043	LDA QUAL ;Get the Qualifier byte
	460E	32 4165	1044	STA LODDST ;Set the load destination
	4611	21 4183	1045	LXI H,CMPTB1 ;Point to this compare table
	4614	22 4162	1046	SHLD CMPADR ;Save it
	4617	3E 80	1047	MVI A,128 ;Bytes in packet
	4619	CD 48C0	1048	CALL TURCV ;Go get the data
			1049	;
			1050	;; If TURCV encounters a non-fatal error, it will return with the
			1051	;; Z bit clear.
			1052	;
461C	C2 4663		1053	JNZ FIXERR ;If error, go fix things up
461F	CD 492F		1054	CALL GETEND ;Go get the END Packet
			1055	;
			1056	;; If GETEND encounters a non-fatal error, if will return with the
			1057	;; Z bit clear.
			1058	;
4622	C2 4663		1059	JNZ FIXERR ;If error, go fix things up
4625	3A 4115		1060	LOAD2X: LDA QUAL ;Check for a P or V load
4628	FE 03		1061	CPI 03 ;Is it?
462A	D2 463A		1062	JNC LOAD3X ;Jump if not
462D	CD 5186		1063	CALL RCVMSX ;Go get the Packet from the CPU
4630	3A 71C6		1064	LDA RCVDAT ;Check the checksum
4633	21 4082		1065	LXI H,BYTSUM ;Point to my byte checksum
4636	BE		1066	CMP M ;Compare them
4637	C2 466B		1067	JNZ FIXER1 ;Jump if not equal
463A	2A 4174		1068	LOAD3X: LHLD BLKNUM ;Get the Starting Block address
463D	11 0010		1069	LXI D,016 ;This is how may blocks we have read
4640	19		1070	DAD D ;Add this to the block address
4641	22 4174		1071	SHLD BLKNUM ;Update the starting block address
4644	3A 4115		1072	LDA QUAL ;Get the qualifier
4647	FE 06		1073	CPI 06 ;See if this is a U load
4649	C8		1074	RZ ;Leave if it is
464A	21 40A3		1075	LXI H,LODADR ;Now we must fix the address
464D	D2 4656		1076	JNC LOAD4X ;Check for WCS load
4650	01 418E		1077	LXI B,XFRCNT ;This is how many bytes to add
4653	C3 6251		1078	JMP ADDAD1 ;Go add them

Error	Addr	Code	Seq	Source statement	
			1079		
4656	2A	41B0	1080	LOAD4X: LHLD	TRKUPC ;Get the updated UPC
4659	22	40A3	1081	SHLD	LODADR ;Save the new value
465C	2A	419A	1082	LHLD	SAVEB0 ;Get the last two bytes of the block
465F	22	419C	1083	SHLD	COPYB0 ;Copy them to here
4662	C9		1084	RET	;leave
			1085		
			1086		
4663	3A	4115	1087	FIXERR: LDA	QUAL ;Get the Qual byte
4666	FE	03	1088	CPI	03 ;Is this a Main Memory load?
4668	DC	46A6	1089	CC	FIXCPU ;Go fixup the CPU if it is
466B	2A	4172	1090	FIXER1: LHLD	PACKS ;Get the Packet count back
466E	22	40A8	1091	SHLD	PAKCNT ;Put it in the Packet Count location
4671	3A	4115	1092	LDA	QUAL ;Get the qual byte
4674	FE	07	1093	CPI	07H ;Is it WCS space
4676	CA	4685	1094	JZ	FIXER2 ;Jump if WCS load
4679	21	40A3	1095	LXI	H,LODADR ;Point to the Load address area
467C	11	418A	1096	LXI	D,XFRADR ;Point to the old address
467F	CD	00F9	1097	CALL	COPY ;Go copy the old address over
4682	C3	45F9	1098	JMP	LOAD1X ;Go try again
4685	3A	4199	1099	FIXER2: LDA	SAVCNT ;Get back the pointer that tell which byte
			1100		;to load
4688	32	4187	1101	STA	COUNT ;And fix the count
468B	2A	419C	1102	LHLD	COPYB0 ;Get the last bytes from the last block
468E	7D		1103	MOV	A,L ;Get the low byte
468F	D3	08	1104	OUT	WCSB0 ;Send it to byte 0
4691	7C		1105	MOV	A,H ;Get the high byte
4692	D3	09	1106	OUT	WCSB1 ;Send it to byte 1
4694	2A	40A3	1107	LHLD	LODADR ;Get the unmodified address
4697	22	41B0	1108	SHLD	TRKUPC ;Restore the Tracking address
469A	22	41AA	1109	SHLD	SWAP ;Get ready to fix the UPC
469D	21	41AA	1110	LXI	H,SWAP ;Point to it
46A0	CD	49D1	1111	CALL	LDUPC ;Go load the UPC
46A3	C3	45F9	1112	JMP	LOAD1X ;Go try again
			1113		
46A6	2A	40A8	1114	FIXCPU: LHLD	PAKCNT ;Get the packet count
46A9	2B		1115	DCX	H ;Reduce it by one
46AA	7C		1116	MOV	A,H ;Put the high byte in A
46AB	B5		1117	ORA	L ;Or in the low byte
46AC	C2	46B3	1118	JNZ	FIXCP1 ;Jump if not zero
46AF	CD	5186	1119	CALL	RCVMSX ;Go get the packet from the CPU
46B2	C9		1120	RET	;Leave
46B3	22	40A8	1121	FIXCP1: SHLD	PAKCNT ;Save it
46B6	06	80	1122	MVI	B,128 ;Count of 128
46B8	CD	514E	1123	CALL	SNDMSG ;Send garbage to the CPU
46BB	C3	46A6	1124	JMP	FIXCPU ;Loop back
			1125		
			1126		
46BE	37		1127	MULSHF: STC	;Set the carry
46BF	3F		1128	CMC	;and clear it

Error	Addr	Code	Seq	Source statement	
	46C0	7D	1129	MULSH1: MOV A,L	;Get the low byte
	46C1	17	1130	RAL	;Rotate it
	46C2	6F	1131	MOV L,A	;Put it back
	46C3	7C	1132	MOV A,H	;Get the high byte
	46C4	17	1133	RAL	;Rotate it
	46C5	67	1134	MOV H,A	;and put it back
	46C6	05	1135	DCR B	;Decrement the count
	46C7	C2 46C0	1136	JNZ MULSH1	;Loop if not done
	46CA	C9	1137	RET	;Leave when done
			1138		
			1139		
			1140		
	46CB	AF	1141	LSTPRP: XRA A	;Make a 0
	46CC	109B	1142	STA LODFLG	;Zero the Load Flag
	46CF	3C	1143	INR A	;Make a 1
	46D0	32 409C	1144	STA DIRFLG	;Set this flag
	46D3	CD 0104	1145	CALL SAVESP	;Go save the stack value
	46D6	CD 0058	1146	CALL TPINI1	;Go init the drives
	46D9	21 0006	1147	LIST: LXI H,6	;Make the Block address 6
	46DC	22 4097	1148	LIST1: SHLD BLOCK	;Put in the Block part of the Packet
	46DF	AF	1149	XRA A	;Make a 0
	46E0	32 417E	1150	STA SEGNUM	;and this counter
	46E3	32 41B4	1151	LIST8: STA LIMIT	;Init this flag
	46E6	2A 42FE	1152	LHLD STRBLK	;Get the starting block value
	46E9	22 4174	1153	SHLD BLKNUM	;Set up this pointer
	46EC	CD 478C	1154	LIST7: CALL GETONE	;Go get a byte
	46EF	CD 478C	1155	CALL GETONE	;Get the one after it
	46F2	FE 08	1156	CPI 8	;Is it End of Segment?
	46F4	CA 476D	1157	JZ ENDSEG	;Jump if it is
	46F7	32 4178	1158	STA ENTYPE	;Otherwise save the Entry type for a moment
	46FA	CD 482B	1159	CALL RAD50	;Go get the file name and extension
	46FD	3A 409B	1160	LDA LODFLG	;See if this is a Load Command
	4700	1F	1161	RAR	;Put the flag in the Carry
	4701	DA 472F	1162	JC LIST4	;Jump if it is
	4704	3A 4178	1163	LDA ENTYPE	;Get the entry type back
	4707	FE 04	1164	CPI 4	
	4709	C2 4710	1165	JNZ LIST2	;Don't print if it is not permanent
	470C	21 41C2	1166	LXI H,LSTBUF	;Point to the Listing Buffer
	470F	F7	1167	RST 6	;Go print it
	4710	01 0006	1168	LIST2: LXI B,06	;Add the 6 bytes we normally skip over
	4713	2A 42FC	1169	LIST3: LHLD EXTRAS	;Get the number of extra words per entry
	4716	CD 4823	1170	CALL MUL2	;Go multiply by two
	4719	09	1171	DAD B	;Add them here
	471A	22 417F	1172	SHLD INCCNT	;Save the count
	471D	CD 478C	1173	BMPPT: CALL GETONE	;Go get a byte
	4720	2A 417F	1174	LHLD INCCNT	;Get the count
	4723	2B	1175	DCX H	;Reduce it
	4724	22 417F	1176	SHLD INCCNT	;Put it back
	4727	7C	1177	MOV A,H	;High byte to A
	4728	B5	1178	ORA L	;OR in the low bye

Error	Addr	Code	Seq	Source statement	
4729	C2 471D	1179	JNZ	BMPPT	;Loop if not done
472C	C3 46EC	1180	JMP	LIST7	;Loop until Endseg with Next Segment=0
		1181			
		1182			
472F	21 41C3	1183	LIST4:	LXI	H,LSTBUF+1 ;Point to where the RAD50 has been unpacked
4732	11 4107	1184		LXI	D,FNBUFF ;Point to the File Name Buffer area
4735	06 0A	1185		MVI	B,10 ;Number of bytes in filename.ext
4737	1A	1186	LIST5:	LDAX	D ;Get a character
4738	BE	1187		CMP	M ;Are they the same?
4739	C2 4759	1188		JNZ	LIST6 ;Jump out if not
473C	23	1189		INX	H ;Bump one pointer
473D	13	1190		INX	D ;and the other
473E	05	1191		DCR	B ;Decrement the count
473F	C2 4737	1192		JNZ	LIST5 ;Loop if not done
4742	3A 4178	1193		LDA	ENTYPE ;Get the entry type
4745	FE 04	1194		CPI	4 ;Is it a Permanent Entry?
4747	C2 4759	1195		JNZ	LIST6 ;Jump if not
474A	CD 484A	1196		CALL	GETTWO ;Get the next two bytes
474D	2A 417A	1197		LHLD	BYTE00 ;Get the two bytes into HL
4750	22 4166	1198		SHLD	LENGTH ;This is how long the file is
4753	21 4164	1199		LXI	H,FNDFLG ;Point to the File Found Flag
4756	36 01	1200		MVI	M,01 ;Set it
4758	C9	1201		RET	;and leave
		1202			
4759	CD 484A	1203	LIST6:	CALL	GETTWO ;Get the next two bytes
475C	2A 417A	1204		LHLD	BYTE00 ;Get the two bytes into HL
475F	EB	1205		XCHG	;Put them in DE
4760	2A 4174	1206		LHLD	BLKNUM ;Get the Starting Block Number
4763	19	1207		DAD	D ;Add them
4764	22 4174	1208		SHLD	BLKNUM ;Save the amount
4767	01 0004	1209		LXI	B,04 ;Want to skip only 4 bytes
476A	C3 4713	1210		JMP	LIST3 ;Go do it
		1211			
		1212			
		1213			
476D	3A 41B4	1214	ENDSEG:	LDA	LIMIT ;Check the limit flag
4770	A7	1215		ANA	A ;Is it set?
4771	C2 4787	1216		JNZ	ENDSE1 ;Jump if it is
4774	2A 42F8	1217		LHLD	NXTSEG ;Get the Next segment number
4777	7D	1218		MOV	A,L ;Put the low byte in the Accumulator
4778	B4	1219		ORA	H ;OR in the high byte
4779	CA 4787	1220		JZ	ENDSE1 ;Jump if 0
477C	2B	1221		DCX	H ;Subtract 1
477D	CD 4823	1222		CALL	MUL2 ;Go multiply value in HL by two
4780	01 0006	1223		LXI	B,06 ;This is the base block number of Directory
4783	09	1224		DAD	B ;Add it to Next Segment
4784	C3 46DC	1225		JMP	LIST1 ;Loop back to a place that will store this
4787	AF	1226	ENDSE1:	XRA	A ;Make a 0
4788	32 409C	1227		STA	DIRFLG ;Zero this flag
478B	C9	1228		RET	;and leave

Error	Addr	Code	Seq	Source statement
			1229	
			1230	
			1231	
478C	21	417E	1232	GETONE: LXI H,SEGNUM ;Point to my Segment Half counter
478F	7E		1233	MOV A,M ;Get it
4790	A7		1234	ANA A ;Check for 0
4791	CA	47AE	1235	JZ GETON2 ;Jump if 0
4794	3A	417D	1236	LDA DIRADR+1 ;Get the high byte of the address
4797	FE	45	1237	CPI 045H ;Is it 45(hex)?
4799	C2	47D7	1238	JNZ GETON4 ;Jump if it isn't
479C	3A	41B4	1239	LDA LIMIT ;Get the limit flag
479F	A7		1240	ANA A ;Is it set?
47A0	CA	47A8	1241	JZ GETON1 ;Jump if not set
47A3	2A	41B5	1242	LHLD LIMSP ;Get the true SP value
47A6	F9		1243	SPHL ;Set up the SP
47A7	C9		1244	RET ;Leave
47A8	7E		1245	GETON1: MOV A,M ;Get my Segment Half counter again
47A9	FE	02	1246	CPI 02 ;See if it is 2
47AB	CC	42A4	1247	CZ TUERR1 ;Error if it is
47AE	21	4300	1248	GETON2: LXI H,DIRBUF ;Set up this address
47B1	22	41B8	1249	SHLD SAVADR ;Save the address
47B4	CD	47E0	1250	CALL GET512 ;Go get half a segment
47B7	21	417E	1251	LXI H,SEGNUM ;Point to my segment-half counter
47BA	34		1252	INR M ;Bump it
47BB	7E		1253	MOV A,M ;Get it
47BC	FE	02	1254	CPI 2 ;Is it now two?
47BE	21	4300	1255	LXI H,DIRBUF ;Point to the buffer (doesn't affect CC's)
47C1	CA	47D4	1256	JZ GETON3 ;Jump if it is
47C4	EB		1257	XCHG ;Move the address into DE
47C5	21	42F6	1258	LXI H,DIRSEG ;Point to this area
47C8	06	0A	1259	MVI B,10 ;Set up a count
47CA	CD	00FB	1260	CALL COPY1 ;Go copy these 10 bytes over
47CD	2A	42FE	1261	LHLD STRBLK ;Get the block where these files start
47D0	22	4174	1262	SHLD BLKNUM ;and put it here where we can work on it
			1263	
			1264	;NOTE: We came back from COPY1 with the correct address in DE
			1265	
47D3	EB		1266	XCHG ;Put DE into HL
47D4	22	417C	1267	GETON3: SHLD DIRADR ;Save the address
47D7	2A	417C	1268	GETON4: LHLD DIRADR ;Get the address
47DA	7E		1269	MOV A,M ;Get the data
47DB	23		1270	INX H ;Bump the address
47DC	22	417C	1271	SHLD DIRADR ;Put the updated address back
47DF	C9		1272	RET ;Leave
			1273	
47E0	21	408C	1274	GET512: LXI H,RETRY ;Point to the retry counter
47E3	36	09	1275	MVI M,9 ;Set up for 8 retries
47E5	2A	41B8	1276	GT512A: LHLD SAVADR ;Get the address
47E8	22	40A3	1277	SHLD LODADR ;Put it here
47EB	21	4165	1278	LXI H,LODDST ;Point to the Load Dest. byte

Error	Addr	Code	Seq	Source statement	
47EE	36 06	1279	MVI	M,06	;Specify Ram
47F0	21 4183	1280	LXI	H,CMPTB1	;Point to this compare table
47F3	22 4162	1281	SHLD	CMADR	;Save it
47F6	21 0200	1282	LXI	H,512	;Set up byte count
47F9	22 4095	1283	SHLD	BYTCNT	;Put it in the Byte Count part of the Packet
47FC	CD 0095	1284	CALL	TU_SEND_PACKET	;Send the Command Packet
47FF	21 0004	1285	LXI	H,04	;Set up the packet count number
4802	22 40A8	1286	SHLD	PAKCNT	;Put it in the packet count
4805	3E 80	1287	MVI	A,128	;Load up the byte count for the Receive routine
4807	CD 48C0	1288	CALL	TURCV	;Go get the segment
480A	C2 47E5	1289	JNZ	GT512A	;If no good, try again
480D	CD 492F	1290	CALL	GETEND	;Go get the End Packet
4810	C2 47E5	1291	JNZ	GT512A	;Try again if error
4813	3E 01	1292	MVI	A,01	;Get ready to set the DIRVAL flag
4815	32 4194	1293	STA	DIRVAL	;Set it
4818	3A 4091	1294	LDA	UNIT	;Get the unit number
481B	32 4195	1295	STA	DIRUNI	;Copy the unit number
481E	21 4097	1296	LXI	H,BLOCK	;Point to the block number
4821	34	1297	INR	M	;Bump it
4822	C9	1298	RET		;Leave
		1299			
		1300			
4823	AF	1301	MUL2: XRA	A	;Clear the carry
4824	7D	1302	MOV	A,L	;Get the low byte
4825	17	1303	RAL		;Rotate it
4826	6F	1304	MOV	L,A	;Put it back
4827	7C	1305	MOV	A,H	;Get the high byte
4828	17	1306	RAL		;Shift it left
4829	67	1307	MOV	H,A	;Put it back
482A	C9	1308	RET		;Leave
		1309			
		1310			
		1311			
482B	21 41C3	1312	RAD50: LXI	H,LSTBUF+1	;Point to area where the first three bytes go
482E	CD 483A	1313	CALL	RAD50A	;Go get the two bytes and convert them
4831	21 41C6	1314	LXI	H,LSTBUF+4	;Point to the area for the second three
4834	CD 483A	1315	CALL	RAD50A	;Do it for the next three
4837	21 41CA	1316	LXI	H,LSTBUF+8	;Area for the Extension
483A	22 4168	1317	RAD50A: SHLD	TMPBUF	;Save the address
483D	CD 484A	1318	CALL	GETTWO	;Go get the two bytes to be converted
4840	2A 417A	1319	LHLD	BYTE00	;Get the data into HL
4843	22 415E	1320	SHLD	REMAIN	;Put them in Remainder
4846	CD 4857	1321	CALL	DIV50	;Go divide by 50(8)
4849	C9	1322	RET		;Leave
		1323			
		1324			
		1325			
484A	CD 478C	1326	GETTWO: CALL	GETONE	;Go get a byte
484D	32 417A	1327	STA	BYTE00	;Store it
4850	CD 478C	1328	CALL	GETONE	;Get another

Error	Addr	Code	Seq	Source statement	
	4853	32 417B	1329	STA BYTE01	;Store it
	4856	C9	1330	RET	;Leave
			1331		
			1332		
	4857	11 F9C0	1333	DIV50: LXI D,DIVSR1	;Get the value of the first Divisor (F9C0H)
	485A	CD 4875	1334	CALL SUBDIV	;Go subtract until underflow
	485D	3A 4160	1335	LDA QUO	;Get the first character
	4860	CD 4895	1336	CALL RADCVT	;Go convert the character
	4863	11 FFD8	1337	LXI D,DIVSR2	;This is the two's complement of 28H
	4866	CD 4875	1338	CALL SUBDIV	;Go divide by 28h
	4869	3A 4160	1339	LDA QUO	;Get the second character
	486C	CD 4895	1340	CALL RADCVT	;Go convert it
	486F	3A 415E	1341	LDA REMAIN	;Get the third character
	4872	C3 4895	1342	JMP RADCVT	;Go convert it
			1343		
			1344		
			1345		
	4875	21 0000	1346	SUBDIV: LXI H,0	;Make a word of zero
	4878	22 4160	1347	SHLD QUO	;Zero the quotient area
	487B	2A 415E	1348	SUBDV1: LHLD REMAIN	;Get the dividend
	487E	7C	1349	MOV A,H	;Get the high half
	487F	E6 80	1350	ANI 080H	;Check for MSB set
	4881	19	1351	DAD D	;This is equivalent to Subtracting 28**2
	4882	C2 4888	1352	JNZ SUBDV2	;Jump if the original number was negative
	4885	7C	1353	MOV A,H	;Get the high byte of result of subtraction
	4886	17	1354	RAL	;Check the carry
	4887	D8	1355	RC	;Leave if underflow
	4888	22 415E	1356	SUBDV2: SHLD REMAIN	;Otherwise, update the Remainder
	488B	2A 4160	1357	LHLD QUO	;Get the quotient
	488E	23	1358	INX H	;Increment it
	488F	22 4160	1359	SHLD QUO	;Save it
	4892	C3 487B	1360	JMP SUBDV1	;Loop
			1361		
			1362		
	4895	FE 1B	1363	RADCVT: CPI 01BH	;What is it?
	4897	CA 48A6	1364	JZ SKP321	;Jump if it is 01BH
	489A	D2 48AB	1365	JNC SKP320	;Jump if not a letter
	489D	A7	1366	ANA A	;Check for 0
	489E	CA 48B7	1367	JZ SKP523	;Jump if it is
	48A1	F6 40	1368	ORI 0404	;Make it ASCII
	48A3	C3 48B7	1369	JMP SKP523	;Jump over the rest
	48A6	3E 24	1370	SKP321: MVI A,024H	;This is the conversion
	48A8	C3 48B7	1371	JMP SKP523	;Jump over rest
	48AB	FE 1C	1372	SKP320: CPI 01CH	;Is it this?
	48AD	C2 48B5	1373	JNZ SKP319	;Jump if not
	48B0	3E 2E	1374	MVI A,02EH	;This is the conversion for this code
	48B2	C3 48B7	1375	JMP SKP523	;Jump over
	48B5	C6 12	1376	SKP319: ADI 012H	;Conversion for numbers
	48B7	2A 4168	1377	SKP523: LHLD TMPBUF	;Get the address back
	48BA	77	1378	MOV M,A	;Send the character to the buffer

Error	Addr	Code	Seq	Source statement	
48BB	23		1379	INX H	;Bump the pointer
48BC	22 4168		1380	SHLD TMPBUF	;Save the address
48BF	C9		1381	RET	;Leave
			1382		
			1383		
			1384		
48C0	32 40B4		1385	TURCV: STA TEMP	;Save the byte count here
48C3	3A 40B4		1386	TURCV3: LDA TEMP	;Get the byte count
48C6	32 40A7		1387	STA LODCNT	;Put it here
48C9	D7		1388	RST 2	;Go get a character
48CA	2A 4162		1389	LHLD CMPADR	;Get address of the appropriate compare table
48CD	BF		1390	CMP M	;Is the Flag Byte good?
48CE	C4 0081		1391	CNZ TUERR3	;Leave if not
48D1	32 409E		1392	STA WRDSUM	;The low byte has to be included in the checksum
48D4	D7		1393	RST 2	;Get the byte count
48D5	2A 4162		1394	LHLD CMPADR	;Get the address of the compare table
48D8	23		1395	INX H	;Bump it
48D9	BE		1396	CMP M	;Is the byte count good?
48DA	C4 0081		1397	CNZ TUERR3	;Leave if not
48DD	32 409F		1398	STA WRDSUM+1	;This byte must also be included in the checksum
48E0	D7		1399	TURCV1: RST 2	;Go look for a character
48E1	21 40A7		1400	LXI H,LODCNT	;Point to the count
48E4	CD 00DE		1401	CALL CHKSUM_WORD	;Go checksum it (data in B when we return)
48E7	3A 4165		1402	LDA LODDST	;Check to see where the data is going
48EA	FE 06		1403	CPI 06	;This is the value for a Load to Ram
48EC	DA 5957		1404	JC MEMLOD	;MEMLOD will do further checking
48EF	C2 4A09		1405	JNZ WCSLOD	;Go to WCSLOD if not less than or equal to 6
48F2	2A 40A3		1406	LHLD LODADR	;Get the Destination address
48F5	7C		1407	MOV A,H	;Get the high byte of the address
48F6	FE 7E		1408	CPI 7EH	;Check for the protected area
48F8	CA 4900		1409	JZ TURCV2	;Don't write the data if address too big
48FB	70		1410	MOV M,B	;Send the data out
48FC	23		1411	INX H	;Bump the pointer
48FD	22 40A3		1412	SHLD LODADR	;Save the address
4900	21 40A7		1413	TURCV2: LXI H,LODCNT	;Point to the count
4903	35		1414	DCR M	;Decrement the count
4904	C2 48E0		1415	JNZ TURCV1	;Loop if not done
4907	D7		1416	RST 2	;Go get the low byte of the checksum
4908	21 409E		1417	LXI H,WRDSUM	;Point to the low byte of the checksum
490B	BE		1418	CMP M	;Check it
490C	C2 4924		1419	JNZ TRYCHK	;Go see about returning
490F	D7		1420	RST 2	;Get the high byte
4910	21 409F		1421	LXI H,WRDSUM+1	;Point to the high byte of the checksum
4913	BE		1422	CMP M	;Is it equal to the calculated
4914	C2 4924		1423	JNZ TRYCHK	;Go see about returning
4917	2A 40A8		1424	LHLD PAKCNT	;Get the packet count
491A	2B		1425	DCX H	;Decrement it
491B	22 40A8		1426	SHLD PAKCNT	;Save the new value
491E	7C		1427	MOV A,H	;Double register decrement doesn't set
491F	B5		1428	ORA L	;Condition codes, so check this way

Error Addr	Code	Seq	Source statement	
4920	C2 48C3	1429	JNZ TURCV3	;Loop if not done
4923	C9	1430	RET	;Leave when done
		1431		
4924	CD 0058	1432	TRYCHK: CALL TPINI1	;Go init the tape drive
4927	21 408C	1433	LXI H,RETRY	;Point to the retry counter
492A	35	1434	DCR M	;Reduce the count
492B	C0	1435	RNZ	;Leave with the Z bit clear to indicate error
492C	CD 0079	1436	CALL TUERR2	;Leave this way if we use up the retries
		1437		
		1438		
492F	21 4165	1439	GETEND: LXI H,LODDST	;Set up the load Dest.
4932	36 06	1440	MVI M,06	;Make the dest the ram
4934	21 4185	1441	LXI H,CMPTB2	;Point to this table
4937	22 4162	1442	SHLD CMPADR	;Save the address
493A	2A 40A3	1443	LHLD LODADR	;Get the low two bytes of the address
493D	E5	1444	PUSH H	;Save them
493E	21 40AA	1445	LXI H,ENDPAK	;Point to the storage for the End Packet
4941	22 40A3	1446	SHLD LODADR	;Store it
4944	21 0001	1447	XI H,01	;Packet count of 1
4947	22 40A8	1448	SHLD PAKCNT	;Store it
494A	3E 0A	1449	MVI A,10	;Bytes in End Packet less checksum
494C	CD 48C0	1450	CALL TURCV	;Go get it
494F	C0	1451	RNZ	;Leave if Z bit clear
4950	21 40AA	1452	LXI H,ENDPAK	;Point to the End Packet
4953	7E	1453	MOV A,M	;Get the first byte
4954	FE 40	1454	CPI 040H	;Is it good?
4956	C4 0079	1455	CNZ TUERR2	;Leave if not
4959	23	1456	INX H	;Point to the Success Code
495A	7F	1457	MOV A,M	;Get it
495B	FE 02	1458	CPI 02	;Is it 0 or 1?
495D	D4 0079	1459	CNC TUERR2	;Leave if it isn't
4960	23	1460	INX H	;Bump pointer
4961	3A 4091	1461	LDA UNIT	;Get the unit number we asked for
4964	BE	1462	CMP M	;Is it correct
4965	C4 0079	1463	CNZ TUERR2	;Error if not
4968	23	1464	INX H	;Point to the next byte
4969	3A 4092	1465	LDA SNDPAK+5	;Get the Switch byte
496C	BE	1466	CMP M	;Is it correct?
496D	C4 0079	1467	CNZ TUERR2	;Leave if not
4970	23	1468	INX H	;Point to the next byte
4971	7E	1469	MOV A,M	;Get it
4972	23	1470	INX H	;Bump pointer
4973	B6	1471	ORA M	;OR in the next byte
4974	C4 0079	1472	CNZ TUERR2	;Error if the last 2 bytes weren't 0
4977	23	1473	INX H	;Point to the low byte of the byte count
4978	3A 4095	1474	LDA BYTCNT	;Get low byte of Command Packet byte count
497B	BE	1475	CMP M	;Are they equal?
497C	C4 0079	1476	CNZ TUERR2	;Leave if not
497F	23	1477	INX H	;Point to the high byte of the byte count
4980	3A 4096	1478	LDA BYTCNT+1	;Get the high byte of Byte Count

Error Addr	Code	Seq	Source statement
4983	BE	1479	CMP M ;Check it
4984	C4 0079	1480	CNZ TUERR2 ;Leave if not the same
4987	23	1481	INX H ;Bump pointer
4988	7E	1482	MOV A,M ;Get next to last byte
4989	23	1483	INX H ;Point to last byte
498A	B6	1484	ORA M ;OR in the last lbyte
498B	C4 0079	1485	CNZ TUERR2 ;Leave if not 0
498E	E1	1486	POP H ;Get the low two bytes of the old address
498F	22 40A3	1487	SHLD LODADR ;Put them back
4992	C9	1488	RET ;Return to caller
		1489	
		1490	
		1491	
		1492	*****
		1493	;
		1494	;
		1495	;
		1496	;
4993	E5	1497	WRTNOP: PUSH H ;Save HL on the stack
4994	21 41A9	1498	LXI H,NOPBUF+2 ;Point to high byte of NOP buffer
4997	36 DB	1499	MVI M,0DBH ;Load high byte of NOP
4999	2B	1500	DCX H ;Decrement the pointer
499A	36 00	1501	MVI M,0H ;Load the middle byte of the NOP
499C	2B	1502	DCX H ;Decrement the pointer
499D	36 15	1503	MVI M,015H ;Load the low byte
499F	CD 49A4	1504	CALL LDCSR ;Go load the CSR
49A2	E1	1505	POP H ;Restore HL
49A3	C9	1506	RET ;Leave
		1507	
		1508	
49A4	22 41A2	1509	LDCSR: SHLD PNTR1 ;Save the address
49A7	06 18	1510	LDCSR1: MVI B,24 ;This is the number of bits to be shifted
49A9	D3 A1	1511	OUT DISPAR ;Turn off parity
49AB	D3 A2	1512	OUT VSHIFT ;Set the V-Bus to shift mode
49AD	D3 38	1513	1\$: OUT CLRCR ;Clear the input to the CSR
49AF	2A 41A2	1514	LHLD PNTR1 ;Get the address of the data to be loaded
49B2	0E 03	1515	MVI C,03 ;Count for shifting
49B4	DB 87	1516	IN CSR23 ;Get the MSB out of the CSR
49B6	1F	1517	RAR ;Put it in the carry
49B7	CD 4A00	1518	CALL LSHIFT ;Go shift 3 bytes
49BA	D2 49BF	1519	JNC 2\$;Jump if no carry from shifting
49BD	D3 39	1520	OUT SETCSR ;Set the input bit
49BF	D3 25	1521	2\$: OUT SETCSK ;Set the CSR clock
49C1	D3 24	1522	OUT CLRCR ;and then clear it
49C3	05	1523	DCR B ;Decrement the count
49C4	C2 49AD	1524	JNZ 1\$;
49C7	D3 A3	1525	OUT VNORML ;Set V-BUS to normal mode
49C9	3A 4197	1526	LDA PARFLG ;Check the parity flag
49CC	1F	1527	RAR ;Is it set?
49CD	D0	1528	RNC ;Leave if not

Error Addr	Code	Seq	Source statement	
49CE	D3 A0	1529	OUT	ENBP
49D0	C9	1530	RET	
		1531	;Turn Parity back on	
		1532	;	
		1533	;.....	
		1534	;	
		1535	;.....	
		1536	;	
		1537	;	
		1538	; LDUPC:	
		1539	;	
49D1	22 4176	1540	LDUPC: SHLD	POINTR
49D4	A7	1541	LDUPC1: ANA	A
49D5	06 0F	1542	MVI	B,15
49D7	0E 02	1543	MVI	C,2
49D9	2A 4176	1544	LHLD	POINTR
49DC	CD 4A00	1545	CALL	LSHIFT
49DF	D3 A2	1546	OUT	VSHIFT
49E1	D3 A8	1547	1\$: OUT	CLRUPC
49E3	DB 84	1548	IN	UPC14
49E5	1F	1549	RAR	
49E6	0E 02	1550	MVI	C,2
49E8	2A 4176	1551	LHLD	POINTR
49EB	CD 4A00	1552	CALL	LSHIFT
49EE	D2 49F3	1553	JNC	2\$
49F1	D3 A9	1554	OUT	SETUPC
49F3	D3 27	1555	2\$: OUT	SETUPK
49F5	D3 26	1556	OUT	CLRUPK
49F7	05	1557	DCR	B
49F8	C2 49E1	1558	JNZ	1\$
49FB	D3 A3	1559	OUT	VNORML
49FD	D3 A9	1560	OUT	CLRACK
49FF	C9	1561	RET	
		1562	;	
		1563	;.....	
		1564	;	
		1565	;.....	
		1566	;	
		1567	;	
		1568	; LSHIFT:	
		1569	;	
4A00	7E	1570	LSHIFT: MOV	A,M
4A01	17	1571	RAL	
4A02	77	1572	MOV	M,A
4A03	23	1573	INX	H
4A04	0D	1574	DCR	C
4A05	C2 4A00	1575	JNZ	LSHIFT
4A08	C9	1576	RET	
		1577	;	
		1578	;.....	

Error	Addr	Code	Seq	Source statement
			1579	
			1580	
			1581	;
			1582	;
			1583	; WCSLOD:
			1584	;
4A09	3A 41B1		1585	WCSLOD: LDA TRKUPC+1 ;Get the high half of the address
4A0C	FE 50		1586	CPI 50H ;Check the high byte of the address
4A0E	D2 4900		1587	JNC TURCV2 ;Jump if not less than 50H
4A11	21 4187		1588	LXI H,COUNT ;Point to the count
4A14	7E		1589	MOV A,M ;Get it
4A15	3C		1590	INR A ;Bump the count
4A16	77		1591	MOV M,A ;Put it back
4A17	FE 02		1592	CPI 02 ;Check for 1, 2 or 3
4A19	78		1593	MOV A,B ;Get the character(doesn't affect CC's)
4A1A	DA 4A36		1594	JC WRBYT0 ;Jump if 1
4A1D	CA 4A3E		1595	JZ WRBYT1 ;Jump if 2
4A20	D3 0A		1596	OUT WCSB2 ;Write WCS register 2
4A22	D3 37		1597	OUT SETWCK ;Clock the U word in
4A24	D3 36		1598	OUT CLRWCK ;Reset the clock
4A26	D3 27		1599	OUT SETUPK ;Clock the UPC
4A28	D3 26		1600	OUT CLRUPK ;Reset the clock
4A2A	AF		1601	XRA A ;Make a 0
4A2B	77		1602	MOV M,A ;Zero the counter
4A2C	2A 41B0		1603	LHLD TRKUPC ;Get the whole address
4A2F	23		1604	INX H ;Bump it
4A30	22 41B0		1605	SHLD TRKUPC ;Save it
4A33	C3 4900		1606	JMP TURCV2 ;Leave
			1607	;
			1608	;
			1609	;
			1610	;
			1611	;
			1612	; WRBYT0
			1613	; WRBYT1
			1614	;
			1615	; Write Byte 0 or 1 of the WCS loading register
			1616	;
4A36	D3 08		1617	WRBYT0: OUT WCSB0 ;Send it to byte 0
4A38	32 419A		1618	STA SAVEB0 ;Save byte 0
4A3B	C3 4900		1619	JMP TURCV2 ;Leave
			1620	
4A3E	D3 09		1621	WRBYT1: OUT WCSB1 ;Write it to byte 1
4A40	32 419B		1622	STA SAVEB1 ;Save byte 1
4A43	C3 4900		1623	JMP TURCV2 ;Leave
			1624	;
			1625	;
			1626	
			1627	
			1628	

Error	Addr	Code	Seq	Source statement
			1629	;*****
			1630	;
4A46	E7		1631	CLLTLK: RST 4 ;Keep things up
4A47	CD 080F		1632	CALL TALK_VECTOR ;Go do the talk stuff
4A4A	C3 4A58		1633	JMP GETLIO ;Skip the next part
			1634	
4A4D	22 40C8		1635	GETLIN: SHLD CNTADR ;Save the address of the count
4A50	23		1636	INX H ;The buffer starts after the count storage area
4A51	22 40CA		1637	SHLD BUFADR ;Save the pointer
4A54	AF		1638	XRA A ;Make a 0
4A55	32 4082		1639	STA BYTSUM ;Zero out the Byte Checksum area
4A58	3A 7E04		1640	GETLIO: LDA TLKFLG ;Get the Talk Flag
4A5B	1F		1641	RAR ;Is it set?
4A5C	DA 4A46		1642	JC CLLTLK ;Jump if it is
4A5F	3A 40BE		1643	LDA LRMASK ;Get the Local/Remote Mask
4A62	FE 81		1644	CPI 81H ;See if Local is set
4A64	C2 4A74		1645	JNZ GETLIA ;Jump if not
4A67	DB 01		1646	IN BOOTSW ;Get the Boot Switch
4A69	1F		1647	RAR ;Put it in the carry
4A6A	DA 4A74		1648	JC GETLIA ;Jump if not asserted
4A6D	3A 408B		1649	LDA CODFLG ;See if the console is loaded
4A70	1F		1650	RAR ;Check the bit
4A71	D2 57E4		1651	JNC BSPREP ;Go try to boot things up
4A74	CD 0046		1652	GETLIA: CALL GS_VECTOR ;Go look for characters via the Silo
4A77	D2 4A58		1653	JNC GETLIO ;Loop if no character found
4A7A	78		1654	GETLI1: MOV A,B ;Get the character
4A7B	E6 7F		1655	ANI 7FH ;Clear the MSB
4A7D	FE 1C		1656	CPI 1CH ;Look for less than 1CH
4A7F	D2 4A91		1657	JNC GETLI4 ;Jump if not
4A82	FE 15		1658	CPI CNTRLU ;Look for ^U
4A84	CA 42CE		1659	JZ CNTLU ;Jump if it is
4A87	FE 12		1660	CPI CNTRLR ;Look for ^R
4A89	CA 42AC		1661	JZ CNTLR ;Jump if it is
4A8C	FE 1B		1662	CPI ALTMOD ;Look for ALT MODE
4A8E	CA 4B2E		1663	JZ ALTCHK ;Jump if it is
4A91	FE 7F		1664	GETLI4: CPI DELETE ;Look for a Delete
4A93	CA 4B0E		1665	JZ RUBOUT ;Jump if it is
4A96	32 40B4		1666	STA TEMP ;Save the character for a while
4A99	3A 40CE		1667	LDA DELFLG ;Get the Delete Flag
4A9C	1F		1668	RAR ;Is it set?
4A9D	DA 4B04		1669	JC SNDEL ;Jump if it is
4AA0	2A 40C8		1670	GETLI2: LHLD CNTADR ;Get the address of the Character Count
4AA3	11 4182		1671	LXI D,CMDVAL ;Point to the Command Valid flag
4AA6	1A		1672	LDAX D ;Get the flag
4AA7	3D		1673	DCR A ;See if it is zero
4AA8	C2 4AAD		1674	JNZ GETLI3 ;Jump if it was
4AAB	12		1675	STAX D ;Make it zero
4AAC	77		1676	MOV M,A ;Zero the count
4AAD	3A 40B4		1677	GETLI3: LDA TEMP ;Get the character back
4AB0	FE 61		1678	CPI 061H ;Check for less than 'a'

Error	Addr	Code	Seq	Source statement	
	4AB2	DA 4ABC	1679	JC	GETLI5 ;Jump if it is
	4AB5	FE 7B	1680	CPI	07BH ;Check for less than '('
	4AB7	D2 4ABC	1681	JNC	GETLI5 ;Jump if not
	4ABA	E6 DF	1682	ANI	LCMASK ;Get rid of Lower Case
	4ABC	32 40B4	1683	GETLI5: STA	TEMP ;Save the character
	4ABF	2A 40CA	1684	LHLD	BUFADR ;Get the address of the buffer
	4AC2	77	1685	MOV	M,A ;Put the character in the buffer
	4AC3	EB	1686	XCHG	;Save HL for a bit
	4AC4	2A 40C8	1687	LHLD	CNTADR ;Get the address of the count
	4AC7	7E	1688	MOV	A,M ;Get the count
	4AC8	FE 50	1689	CPI	MAXCNT ;Check for max
	4ACA	CA 4AD3	1690	JZ	GETLI6 ;Jump if at max
	4ACD	34	1691	INR	M ;Otherwise, bump the count
	4ACE	EB	1692	XCHG	;Restore HL
	4ACF	23	1693	INX	H ;Bump the Buffer address
	4AD0	22 40CA	1694	SHLD	BUFADR ;Save it
	4AD3	3A 40BE	1695	GETLI6: LDA	LRMASK ;Get the Local/Remote Mask
	4AD6	E6 40	1696	ANI	40H ;See if this is an APT character
	4AD8	C2 4AF6	1697	JNZ	GETLI8 ;Jump if APT
	4ADB	3A 40B4	1698	LDA	TEMP ;Get the character back
	4ADE	FE 20	1699	CPI	20H ;See if it is unrecognized control char
	4AE0	D2 4AF5	1700	JNC	GETLI7 ;Jump if it isn't
	4AE3	FE 0D	1701	CPI	CR ;Is it a Carriage Return?
	4AE5	CA 4AF5	1702	JZ	GETLI7 ;Jump if it is
	4AE8	FE 0A	1703	CPI	LF ;Is it a Line Feed?
	4AEA	CA 4AF5	1704	JZ	GETLI7 ;Jump if it is
	4AED	3E 5E	1705	MVI	A,'^' ;Get the Up Arrow
	4AEF	DF	1706	RST	3 ;Go send it out
	4AF0	3A 40B4	1707	LDA	TEMP ;Get the character
	4AF3	F6 40	1708	ORI	40H ;Set the bit to convert the character
	4AF5	DF	1709	GETLI7: RST	3 ;This calls the SNDCHR routine
	4AF6	3A 40B4	1710	GETLI8: LDA	TEMP ;Get the character
	4AF9	FE 0D	1711	CPI	CR ;See if it was a carriage return
	4AFB	C2 4A58	1712	JNZ	GETLI0 ;If not, go look for the next character.
	4AFE	3E 01	1713	MVI	A,1 ;Make a one
	4B00	32 4182	1714	STA	CMDVAL ;Set this flag
	4B03	C9	1715	RET	;and leave
			1716		
			1717		
	4B04	AF	1718	SNDDEL: XRA	A ;Make 0
	4B05	32 40CE	1719	STA	DELFLG ;Clear it if it was set
	4B08	3E 5C	1720	MVI	A,'\' ;Get a backslash
	4B0A	DF	1721	RST	3 ;Go print it
	4B0E	C3 4AA0	1722	JMP	GETLI2 ;Go rejoin the main flow
			1723		
			1724		
	4B0E	2A 40C8	1725	RUBOUT: LHLD	CNTADR ;Get the address of the count
	4B11	AF	1726	XRA	A ;Make a 0
	4B12	BE	1727	CMP	M ;See if the count is already 0
	4B13	CA 4A58	1728	JZ	GETLI0 ;Jump if it is

Error Addr	Code	Seq	Source statement
------------	------	-----	------------------

4B16	35	1729	DCR M ;Reduce the count by one
4B17	21 40CE	1730	LXI H,DELFLG ;Point to the Delete Flag
4B1A	BE	1731	CMP M ;See if the delete flag is set
4B1B	C2 4B22	1732	JNZ RUBOU1 ;If set, don't print another backslash
4B1E	34	1733	INR M ;Set the flag
4B1F	3E 5C	1734	MVI A,'\' ;Get a backslash
4B21	DF	1735	RST 3 ;Go print it
4B22	2A 40CA	1736	RUBOU1: LHLD BUFADR ;Point to the Buffer
4B25	2B	1737	DCX H ;Decrement the pointer
4B26	22 40CA	1738	SHLD BUFADR ;Save the new value
4B29	7E	1739	MOV A,M ;Get the character
4B2A	DF	1740	RST 3 ;Go print it
4B2B	C3 4A58	1741	JMP GETLIO ;Then rejoin the GETLIN flow

4B2E	3A 4182	1746	ALTCHK: LDA CMDVAL ;Check the command valid flag
4B31	A7	1747	ANA A ;If set, this is the first character on line
4B32	3E 0D	1748	MVI A,CR ;Put a Carriage Return in A just in case
4B34	CA 4ABC	1749	JZ GETLIS ;Jump if not set
4B37	2A 40C8	1750	LHLD CNTADR ;Get the pointer to the last command
4B3A	F7	1751	RST 6 ;Go print it
4B3B	C9	1752	RET ;Leave

```

1753
1754
1755 ; .....
1756 ;
1757 ; This is the routine that decides how to make the transition from CONSOL.EXE
1758 ; to ENKAA.EXE (aka MICMON). It is a little tricky (I was pressed for space)
1759 ; so I will explain it step by step.
1760 ;
1761 ; 1) First I get the CODFLG. Note that if CONSOL is resident, CODFLG
1762 ; will equal 0, and if ENKAA is resident it will equal 1. Since ENKAA
1763 ; should be found on Drive 0, and CONSOL should be found onDrive 1,
1764 ; flipping this flag and using it for the Unit number for reading in the
1765 ; new code will work, and we don't have to test the state of the bit at
1766 ; all.
1767 ;
1768 ; 2) Next I test for either Modem In Progress (MIPFLG) OR CRD (either
1769 ; Auto or Menu mode, it doesn't matter). If neither is true, then we put
1770 ; the flipped CODFLG in the Stack Pointer and jump down into the ROM to
1771 ; go through the Self Test and come up from scratch. The LSB of the SP is
1772 ; the only state saved through the Self Test.
1773 ;
1774 ; 3) By setting the OFLAG using the CRD_FLAGS, we save testing for what
1775 ; is going on. Only the LSB of the OFLAG byte is tested by those routines
1776 ; that check it.
1777 ;
1778 ;

```

Error	Addr	Code	Seq	Source statement
-------	------	------	-----	------------------

4B3C			1779	CRD_RD_TEST:
4B3C	3E 06		1780	MVI A,6 ;We have to set up the correct address space in
4B3E	32 4115		1781	STA QUAL ;the QUAL byte.
4B41	21 4C00		1782	LXI H,4C00H ;Get this address
4B44	22 4111		1783	SHLD SBUFFR ;Store it here
4B47	21 408B		1784	LXI H,CODFLG ;Point to this flag
4B4A	7E		1785	MOV A,M ;Get it into A
4B4B	32 4091		1786	STA UNIT ;Set up the TU-58 Unit Number
4B4E	EE 01		1787	XRI 1 ;Flip the Code Flag
4B50	77		1788	MOV M,A ;Put the new value back.
4B51	5F		1789	MOV E,A ;Also put it in E.
4B52	AF		1790	XRA A ;Make a 0
4B53	57		1791	MOV D,A ;Zero D.
4B54	32 4080		1792	STA RAMVAL ;Clear out the Ram Valid flag.
4B57	3A 40FD		1793	LDA CRD_FLAGS ;Point to the CRD Flags byte
4B5A	32 40BB		1794	STA OFLAG ;Set the Control 0 Flag accordingly.
4B5D	21 40D9		1795	LXI H,MIPFLG ;Point to the Modem In Progress Flag.
4B60	B6		1796	ORA M ;OR them together
4B61	C2 420C		1797	JNZ MYLOAD ;Go to the MYLOAD entry if (MIP OR CRD).
4B64	EB		1798	XCHG ;Otherwise, put DE in HL,
4B65	F3		1799	DI ;turn off the interrupts,
4B66	F9		1800	SPHL ;and jam HL into the Stack Pointer.
4B67	C3 0004		1801	JMP BEGIN+4 ;Go do the self test and destroy almost
			1802	everything.

			1803	;
			1804	;.....
			1805	
			1806	
			1807	
			1808	
			1809	
	=4B80		1810	ORG 4B80H ;
			1811	
4B80	=0040		1812	CHRSIL: DS 64 ;64 byte character silo
4BC0	=0040		1813	SRCSIL: DS 64 ;64 byte character source silo
			1814	;
			1815	;.....
			1816	
			1817	
			1818	
			1819	
			1820	
			1821	;#####
			1822	;
			1823	;Starting at 4C00H we can have either CONSOL.EXE or MICHON.EXE
			1824	;
			1825	;.....
			1826	;
	=4C00		1827	ORG 04C00H ;
			1828	

Error	Addr	Code	Seq	Source statement
	4C00		1829	GO_START:
	4C00	AF	1830	XRA A ;Make a 0
	4C01	32 40D5	1831	STA WCS_LOAD_FLAGS ;Clear both flags.
	4C04	21 729D	1832	LXI H,VERMSG ;Point to this message
	4C07	F7	1833	RST 6 ;Go print it
	4C08	3A 40D6	1834	LDA APTLOD ;See if we got here via X command
	4C0B	1F	1835	RAR ;Is the APTLOD flag set?
	4C0C	D2 5F99	1836	JNC COLD_BEGIN ;Jump if not
	4C0F	C3 4C32	1837	JMP IDLE ;Otherwise, go to the IDLE LOOP
			1838	
			1839	
		=4C22	1840	ORG 94C22H ;
			1841	
	4C22	C3 5E56	1842	PARVEC: JMP PARITY ;
	4C25	C3 5DE5	1843	ACVECT: JMP ACFAIL ;
	4C28	C3 4C32	1844	CVECTR: JMP IDLE ;
	4C2B	C3 4C32	1845	PVECTR: JMP IDLE ;
	4C2E	00	1846	CODFL1: DB 0 ;Flag to be used when X/U 4100 brings in
			1847	;MICRON or CONSOL
	4C2F	C3 6A18	1848	TMRVEC: JMP TMRSRV ;Go service the I.T. overflow
			1849	
			1850	;*****
			1851	;*****
			1852	
			1853	;The following is the Nebula Console software. Overall this code consists of
			1854	;an input routine that brings in a command string, routines that implement
			1855	;each command, a parser that, if called, pigeonholes information about the
			1856	;command string for use by the calling command, and various routines that:
			1857	;(1) manipulate addresses and data,
			1858	;(2) handle errors,
			1859	;(3) communicate with the CPU, the Console terminal and the TU-58,
			1860	;(4) handle Power Up, Power Down and Boot functions.
			1861	;
			1862	;
			1863	
			1864	
			1865	
			1866	
			1867	
			1868	
			1869	;The following are maps of the Flag and Qual bytes:
			1870	;
			1871	;
			1872	; 07 06 05 04 03 02 01 00
			1873	;
			1874	;FLAGS DACTIV AACTIV N.U. H.U. N.U. DFLAG AFLAG SPCFLG
			1875	;
			1876	;FLAGS1 N.U. N.U. N.U. N.U. * - +
			1877	;
			1878	;FLAGS2 N.U. N.U. N.U. N.U. N.U. N.U. NACTIV NFLAG

Error	Addr	Code	Seq	Source statement
			1879	;
			1880	;FLAGS3 N.U. N.U. N.U. N.U. N.U. N.U. SACTIV SFLAG
			1881	;
			1882	;OFLAG N.U. N.U. N.U. N.U. N.U. N.U. N.U. O FLAG
			1883	;
			1884	;DELFLG N.U. N.U. N.U. N.U. N.U. N.U. N.U. DELFLG
			1885	;
			1886	;LODFLG N.U. N.U. N.U. N.U. N.U. N.U. N.U. LOAD FLAG
			1887	;
			1888	;RFLAG N.U. N.U. N.U. N.U. N.U. N.U. N.U. R FLAG
			1889	;
			1890	;QUAL (D.L. CODE) N.U. N.U. (ADDRESS SPACE CODE)
			1891	;
			1892	
			1893	
			1894	
			1895	
			1896	
			1897	
			1898	
			1899	*****
			1900	;
			1901	; CNSOLE
			1902	;
			1903	; The following is the Console Mode idle loop and the normal entry
			1904	; point to it.
			1905	;
4C32	31	4080	1906	IDLE: LXI SP,STACK ;Fix up the Stack Pointer
4C35	3A	409D	1907	LDA INDFLG ;Check the Indirect Command flag
4C38	1F		1908	RAR ;Put it in the carry
4C39	D4	4C42	1909	CNC CNSOLA ;If not set use the normal console code
4C3C	DC	4C5E	1910	CC CNSOLB ;If set use the special code
4C3F	C3	4C32	1911	JMP IDLE ;Loop back
			1912	
4C42	AF		1913	CNSOLA: XRA A ;Make a zero
4C43	32	40BB	1914	STA OFLAG ;Zero this flag
4C46	CD	4CCF	1915	CALL PRPSUB ;Go prep some flags
4C49	21	4203	1916	LXI H,PROMPT ;Point to the prompt
4C4C	F7		1917	MVI 6 ;Go print it
4C4D	21	7000	1918	LXI H,CHRCNT ;Point to the Character Count location
4C50	CD	4A4D	1919	CALL GETLIN ;Go get a line
4C53	21	7001	1920	LXI H,CHRBUF ;Point to the character buffer
4C56	22	40CA	1921	SHLD BUFADR ;Save the address
4C59	CD	4C7B	1922	CALL CNSOLX ;Go use a common subroutine
4C5C	AF		1923	XRA A ;Clear the carry
4C5D	C9		1924	RET ;and go back to the idle loop
			1925	
4C5E	CD	4C94	1926	CNSOLB: CALL INDTST ;Go check the Indirect Command buffer
4C61	C8		1927	RZ ;Leave if we ran out
4C62	CD	4CCF	1928	CALL PRPSUB ;Go prep some flags

Error	Addr	Code	Seq	Source statement	
	4C45	21 4203	1929	LXI H,PROMPT	:Point to the prompt
	4C68	F7	1930	RST 6	:Go print it
	4C69	3A 721B	1931	LDA INDCNT	:Get the count
	4C6C	2A 40CA	1932	LHLD BUFADR	:Get the address
	4C6F	EF	1933	RST 5	:Special entry to Send Line
	4C70	CD 4C7B	1934	CALL CNSOLX	:Go use this subroutine
	4C73	3A 4116	1935	LDA SUCCES	:Check the success code
	4C76	A7	1936	ANA A	:Is it zero?
	4C77	CA 4C5E	1937	JZ CNSOLB	:Loop back
	4C7A	C9	1938	RET	:Leave if non-zero
			1939		
	4C7B	CD 5DB1	1940	CNSOLX: CALL MARKSP	:Go mark where we came from
	4C7E	2A 40CA	1941	LHLD BUFADR	:Get the address of the command buffer
	4C81	CD 4D23	1942	CALL SPSTRP	:Go strip off the spaces
	4C84	23	1943	INX H	:Bump the pointer beyond the command
	4C85	22 40CA	1944	SHLD BUFADR	:Save the address
	4C88	21 7101	1945	LXI H,COUNT1	:Point to the Compare table
	4C8B	11 7114	1946	LXI D,ADRES1	:Point to the Address Table
	4C8E	CD 4CFA	1947	CALL DECODE	:Go figure it out
	4C91	CD 5D1E	1948	CALL SYNERR	:Error if no match
			1949		
	4C94	06 00	1950	INDTST: MVI B,0	:Zero this counter
	4C96	2A 721C	1951	LHLD INDADR	:Get the current pointer value
	4C99	22 40CA	1952	SHLD BUFADR	:Save it
	4C9C	CD 4CC6	1953	CALL GETCHK	:Get the first character in this line
	4C9F	A7	1954	ANA A	:Is it zero?
	4CA0	C2 4CA7	1955	JNZ INDTS1	:Jump if not zero
	4CA3	32 409D	1956	STA INDFLG	:Zero this flag
	4CA6	C9	1957	RET	:Leave with the Z bit set
	4CA7	FE 0D	1958	INDTS1: CPI CR	:See if it is a Carriage Return
	4CA9	CA 4CB6	1959	JZ INDTS2	:Jump if it is
	4CAC	04	1960	INR B	:Bump the count
	4CAD	FC 5D1E	1961	CM SYNERR	:If minus, the count is G.T. 128
	4CB0	CD 4CC6	1962	CALL GETCHK	:Go bump the pointer and check it
	4CB3	C3 4CA7	1963	JMP INDTS1	:Loop back
	4CB6	7E	1964	INDTS2: MOV A,M	:Get the next character
	4CB7	FE 0A	1965	CPI LF	:Is it a line feed
	4CB9	C4 5D1E	1966	CNZ SYNERR	:Error if not
	4CBC	23	1967	INX H	:Bump the pointer
	4CBD	22 721C	1968	SHLD INDADR	:Save the new address
	4CC0	78	1969	MOV A,B	:Put the count in A
	4CC1	3C	1970	INR A	:Bump it to count the CR but not the LF
	4CC2	32 721B	1971	STA INDCNT	:Save it
	4CC5	C9	1972	RET	:Return to do the command
			1973		
	4CC6	7C	1974	GETCHK: MOV A,H	:Get the high half of the address
	4CC7	FE 7E	1975	CPI 7EH	:Has it gone out of range?
	4CC9	CC 5D16	1976	CZ CF_ERROR	:Error if set
	4CCC	7E	1977	MOV A,M	:Put the data in A
	4CCD	23	1978	INX H	:Bump the pointer

Error	Addr	Code	Seq	Source statement
	4CCE	C9	1979	RET ;Leave
			1980	
			1981	
	4CCF	AF	1982	PRPSUB: XRA A ;Make a zero
	4CD0	32 7213	1983	STA FLAGS ;Start zeroing things out
	4CD3	32 7214	1984	STA FLAGS1 ;
	4CD6	32 7215	1985	STA FLAGS2 ;
	4CD9	32 7216	1986	STA FLAGS3 ;
	4CDC	32 7217	1987	STA RFLAG ;
	4CDF	32 7218	1988	STA QUAL1 ;
	4CE2	32 4115	1989	STA QUAL ;
	4CE5	32 409B	1990	STA LODFLG ;
	4CE8	32 721F	1991	STA MICFLG ;
	4CEB	32 721E	1992	STA STPFLG ;
	4CEE	32 71B3	1993	STA NEWAS ;
	4CF1	32 71B2	1994	STA NEWDL ;
	4CF4	32 7223	1995	STA IFLAG ;
	4CF7	D3 AF	1996	OUT CLRHLT ;Turn off the Halt to the CPU
	4CF9	C9	1997	RET ;Leave
			1998	;
			1999	;
			2000	;.....
			2001	;
			2002	;.....
			2003	;
			2004	;
			2005	;
	4CFA	46	2006	DECODE: MOV B,M ;Get the count
	4CFB	23	2007	DECOD1: INX H ;Bump the Compare Table pointer
	4CFC	BE	2008	CMP H ;The accumulator contains the character to be
			2009	;decoded
	4CFD	CA 4D07	2010	JZ MATCH ;If a match, then jump
	4D00	13	2011	INX D ;Bump the Address Table pointer
	4D01	13	2012	INX D ;Twice
	4D02	05	2013	DCR B ;Decrement the Count
	4D03	C8	2014	RZ ;If 0, return
	4D04	C3 4CFB	2015	JMP DECOD1 ;Else, repeat the loop
			2016	
	4D07	E1	2017	MATCH: POP H ;Since we are not going to execute a return
			2018	;we must fix the Stack
	4D08	1A	2019	LDAX D ;Get the low half of the address
	4D09	6F	2020	MOV L,A ;Put it in L
	4D0A	13	2021	INX D ;Point to the high half of the address
	4D0B	1A	2022	LDAX D ;Get the high half of the address
	4D0C	67	2023	MOV H,A ;Put it in H
	4D0D	E9	2024	PCHL ;Exit, stage left
			2025	;
			2026	;.....
			2027	;
			2028	

Error Addr	Code	Seq	Source statement
		2029	
		2030	;The REPEAT flows make sure that a Repeat command is not followed by a
		2031	;Repeat command, and that a Repeat command is seperated from the following
		2032	;command by at least one space.
		2033	
4D0E	21 7217	2034	REPEAT: LXI H,RFLAG ;Point to the RFLAG
4D11	35	2035	DCR M ;Check for already set
4D12	36 01	2036	MVI M,1 ;Set it in case it wasn't
4D14	CC 5D1E	2037	CZ SYNERR ;If it was set, then we have R followed by R.
4D17	2A 40CA	2038	LHLD BUFADR ;Else, check to see that at least one space
4D1A	7E	2039	MOV A,M ;trails the R command
4D1B	FE 20	2040	CPI SPACE ;Is there a space after the R?
4D1D	CA 4C7B	2041	JZ CNSOLX ;Jump back if there is
4D20	CD 5D1E	2042	CALL SYNERR ;If not, error
		2043	
		2044	
		2045	;*****
		2046	;
		2047	; SPSTRP
		2048	;
		2049	; This subroutine searches the buffer pointed to by HL until something
		2050	; otherthan a space is found. It then returns with the character in A and HL
		2051	; still pointing to the memory location where it found the character
		2052	;
4D23	7E	2053	SPSTRP: MOV A,M ;Get the character
4D24	FE 20	2054	CPI SPACE ;Is it a space?
4D26	C0	2055	RNZ ;If not a space, return.
4D27	23	2056	INX H ;Else, bump the pointer and keep looking
4D28	C3 4D23	2057	JMP SPSTRP ;
		2058	;
		2059	;*****
		2060	
		2061	
		2062	
		2063	
		2064	
		2065	;*****
		2066	;
		2067	; PARSE
		2068	;
		2069	;
4D2B	21 71FE	2070	PARSE: LXI H,ABUFFR ;Point to the area to be cleared
4D2E	16 08	2071	MVI D,08 ;Number of bytes to be cleared
4D30	CD 6AFC	2072	CALL ZBUF1 ;Go do it
4D33	3A 409B	2073	PARSE1: LDA LODFLG ;Check for the Load flag
4D36	1F	2074	RAR ;Put it in the carry
4D37	CD 4FC5	2075	CALL GETNXT ;Get the next character
4D3A	21 715D	2076	LXI H,COUNT3 ;HL gets the Compare Table pointer
4D3D	11 716A	2077	LXI D,ADRES3 ;DE gets the Address Table pointer
4D40	D2 4D50	2078	JNC PARSE2 ;Skip the next section if not a Load

Error Addr	Code	Seq	Source statement
------------	------	-----	------------------

4D43	FE 2E	2179	CPI '.'
4D45	CA 4E3B	2180	JZ DOT
4D48	06 05	2091	MVI B,5
4D4A	CD 4CFB	2082	CALL DECOD1
4D4D	C3 4D53	2083	JMP PARSE3
4D50	CD 4CFA	2084	PARSE2: CALL DECODE
4D53	5F	2085	PARSE3: MOV E,A
		2086	
		2087	
		2088	
		2089	
		2090	
		2091	
		2092	
4D54	01 7213	2093	LXI B,FLAGS
4D57	0A	2094	LDAX B
4D58	1F	2095	RAR
4D59	D4 5D1E	2096	CNC SYNERR
4D5C	1F	2097	RAR
4D5D	DA 4D71	2098	JC PARSE4
		2099	
4D60	3A 409B	2100	LDA LODFLG
4D63	1F	2101	RAR
4D64	DA 4D8D	2102	JC FNFLOW
4D67	21 71FE	2103	LXI H,ABUFFR
4D6A	0A	2104	LDAX B
4D6B	F6 40	2105	ORI AACTIV
4D6D	02	2106	STAX B
4D6E	C3 4D83	2107	JMP PARSE5
4D71	1F	2108	PARSE4: RAR
4D72	DC 5D1E	2109	CC SYNERR
		2110	
4D75	3A 409B	2111	LDA LODFLG
4D78	1F	2112	RAR
4D79	DA 4DDA	2113	JC EXTFLW
4D7C	21 7202	2114	LXI H,DBUFFR
4D7F	0A	2115	LDAX B
4D80	F6 80	2116	ORI DACTIV
4D82	02	2117	STAX B
4D83	7B	2118	PARSE5: MOV A,E
4D84	CD 4DF4	2119	CALL CNVERT
4D87	CD 4E0D	2120	CALL SHIFT
4D8A	C3 4D33	2121	JMP PARSE1
		2122	
4D8D	0A	2123	FNFLOW: LDAX B
4D8E	E6 40	2124	ANI AACTIV
4D90	C2 4DC0	2125	JNZ FNFLW1
4D93	3E 44	2126	MVI A,'D'
4D95	BB	2127	CMP E
4D96	C2 4DC0	2128	JNZ FNFLW1

```

;Is the character a Dot?
;Go to a special handler if it is.
;Otherwise change the count
;Go figure things out
;Jump over the other call
;Go decode the character.
;If the character being decoded is not
;something special, like a +, or an S,
;then it is assumed that it should be
;a Hex digit. Therefore it is either
;part of the address or the data. Only one
;address field and one data field
;are allowed. The character is saved in
;D while some testing is performed.
;Load BC with the FLAGS address.
;Get the Flags
;Check the Space Flag
;If not set, error.
;Else, check the A Flag
;If the Address field has been terminated, go
;check the Data field.
;Check for Load flag
;Put it in carry
;If Load, go to the File name flow
;Load HL to point to the Address Buffer
;Get the Flags again
;Set the Address Active Flag
;Store it away
;Go share some code
;Check the D Flag
;If this field has been terminated then no
;more data can be accepted.
;Going to check the Load Command Flag
;Put it in the carry
;If set, go to the Extension Flow
;Else, not done with the Data Field yet
;Get the Flags
;Set the Data Active flag
;Store it
;A gets the character again
;Convert ASCII to Hex
;Go shift in the digit
;Loop back

;Check the AACTIV Flag
;Is it set?
;Jump if it is
;Get the ASCII for D
;Is the character a D?
;Jump if not

```

Error	Addr	Code	Seq	Source statement	
	4D99	2A 40CA	2129	LHLD BUFADR	;Get the current address
	4D9C	BE	2130	CMP M	;Check the next letter
	4D9D	C2 4DC0	2131	JNZ FNFLW1	;Jump if not equal
	4DA0	23	2132	INX H	;Bump the pointer
	4DA1	56	2133	MOV D,M	;Put the next character in D for a moment
	4DA2	23	2134	INX H	;and Bump the pointer again
	4DA3	3E 3A	2135	MVI A,':'	;Put value for colon in A
	4DA5	BE	2136	CMP M	;Check it
	4DA6	C2 4DC0	2137	JNZ FNFLW1	;Jump if not equal
	4DA9	7A	2138	MOV A,D	;Get the device number
	4DAA	CD 4DF4	2139	CALL CNVERT	;Go change from Hex to ASCII
	4DAD	FE 02	2140	CPI 02	;Check it
	4DAF	D4 5D1E	2141	CNC SYNERR	;Leave if it isn't 0 or 1
	4DB2	32 4106	2142	STA UNITX	;Store it
	4DB5	23	2143	INX H	;and bump the pointer
	4DB6	22 40CA	2144	SHLD BUFADR	;now update the saved address
	4DB9	21 7212	2145	LXI H,DEVFLG	;Point to this flag
	4DBC	34	2146	INR M	;Set it
	4DBD	C3 4DD3	2147	JMP FNFLW2	;Go set the Address Active flag
	4DC0	21 7210	2148	FNFLW1: LXI H,FNCNT	;Point to the Count
	4DC3	7E	2149	MOV A,M	;Get it
	4DC4	FE 06	2150	CPI 06	;Is the File Name Buffer full?
	4DC6	CC 5D1E	2151	CZ SYNERR	;Leave if it is
	4DC9	3C	2152	INR A	;Otherwise, bump the count
	4DCA	77	2153	MOV M,A	;and store it
	4DCB	2A 720C	2154	LHLD FNPTR	;Get the Address
	4DCE	73	2155	MOV M,E	;Store the character
	4DCF	23	2156	INX H	;Bump the Address
	4DD0	22 720C	2157	SHLD FNPTR	;And save this new value
	4DD3	0A	2158	FNFLW2: LDAX B	;Get the Flags byte
	4DD4	F6 40	2159	ORI AACTIV	;Set the AACTIV flag
	4DD6	02	2160	STAX B	;and put it back
	4DD7	C3 4D33	2161	JMP PARSE1	;Go back to the Parser
			2162		
	4DDA	21 7211	2163	EXTFLW: LXI H,EXTCNT	;Point to the Extension Count
	4DDD	7F	2164	MOV A,M	;and get it
	4DDE	FE 03	2165	CPI 03	;See if the Extension Buffer is full
	4DE0	CC 5D1E	2166	CZ SYNERR	;Leave if it is
	4DE3	3C	2167	INR A	;Otherwise bump the count
	4DE4	77	2168	MOV M,A	;and store it
	4DE5	2A 720E	2169	LHLD EXTPTR	;Get the pointer to the Extension Buffer
	4DE8	73	2170	MOV M,E	;Store the character
	4DE9	23	2171	INX H	;Bump the pointer
	4DEA	22 720E	2172	SHLD EXTPTR	;and save the new value
	4DED	0A	2173	LDAX B	;Get the flags
	4DEE	F6 80	2174	ORI DACTIV	;Set Data Active flag
	4DF0	02	2175	STAX B	;and put it back
	4DF1	C3 4D33	2176	JMP PARSE1	;Go back to the parser
			2177		
			2178	;.....	

Error Addr Code Seq Source statement

```

2179
2180
2181
2182
2183
2184 ;*****
2185 ;
2186 ;       CNVERT
2187 ;
2188 ;       CNVERT changes ASCII representations of Hex into true Hex (Binary)
2189 ; digits.  If the character to be converted is not in the range of Hex
2190 ; digits (ASCII), an error occurs.
2191 ;
4DF4 FE 30 2192 CNVERT: CPI 030H ;Is the character less than 0.
4DF6 DC 5D1E 2193 CC SYNERR ;If it is, error
4DF9 FE 3A 2194 CPI 03AH ;Else, check to see if is in the range
2195 ; from 0 to 9.
4DFB DA 4E0A 2196 JC CNVER1 ;If it is, no convert necessary.
4DFE FE 41 2197 CPI 041H ;Else, see if less than letter A
4E00 DC 5D1E 2198 CC SYNERR ;If it is, error
4E03 FE 47 2199 CPI 047H ;Else, see if it is less than letter G
4E05 D4 5D1E 2200 CNC SYNERR ;If it is not, error
4E08 C6 09 2201 ADI 09H ;Else, turn ASCII A-F into Hex
4E0A E6 0F 2202 CNVER1: ANI 0FH ;Clear the high nibble
4E0C C9 2203 RET ;Adios
2204 ;
2205 ;*****
2206 ;
2207 ;*****
2208 ;
2209 ;       SHIFT
2210 ;
2211 ;       SHIFT is a routine to shift a Hex digit into the low nibble of a buffer
2212 ; area.  It expects the address of the buffer to be in HL and saves this
2213 ; address on the Stack before it does anything.  When it is done it restores
2214 ; HL with the original address.
2215 ;
4E0D 5F 2216 SHIFT: MOV E,A ;Save the digit for a while.
4E0E 06 04 2217 MVI B,04 ;Count for the number of bits to be shifted in.
4E10 0E 04 2218 SHIFT1: MVI C,04 ;Count for number of bytes to shift accross
4E12 E5 2219 PUSH H ;Save the address on the Stack
4E13 AF 2220 XRA A ;This will clear the carry
4E14 CD 4A00 2221 CALL LSHIFT ;Go do the shift
4E17 E1 2222 POP H ;Restore HL
4E18 05 2223 DCR B ;Decrement the bit count.
4E19 C2 4E10 2224 JNZ SHIFT1 ;If not zero, do it again.
4E1C 7B 2225 MOV A,E ;Get the digit into the accumulator.
4E1D B6 2226 ORA M ;OR the digit into low nibble of low byte
4E1E 77 2227 MOV M,A ;Store it.
4E1F C9 2228 RET ;Leave.

```

Error Addr	Code	Seq	Source statement
		2229	;
		2230	;.....
		2231	
		2232	
		2233	;.....
		2234	;
		2235	; DLMCHK
		2236	;
		2237	; DLMCHK (Delimiter Check) is used to set the A and D Flags. If a
		2238	; delimiter is encountered and the Data Active Flag is set, the D(ata) Flag
		2239	; is set. If the D Active Flag is not set but the A Active is set, the
		2240	; A(ddress) Flag is set. If neither Active Flag is set, nothing happens.
		2241	;
4E20	01 7213	2242	DLMCHK: LXI B,FLAGS ;Point to the Flags
4E23	0A	2243	LDAX B ;Read in the Flags
4E24	17	2244	RAL ;Check the Data Active Flag
4E25	D2 4E2D	2245	JNC ACHECK ;If not set, go check the Address Active
		2246	;Flag
4E28	0A	2247	DCHK1: LDAX B ;Else, get a fresh copy of the Flags
4E29	F6 04	2248	ORI DFLAG ;Set the Data Flag which indicates the Data
		2249	;Field is terminated.
4E2B	02	2250	STAX B ;Store it
4E2C	C9	2251	RET ;We are done.
		2252	
		2253	
4E2D	17	2254	ACHECK: RAL ;Check the Address Active Flag
4E2E	D0	2255	RNC ;If not set, return.
4E2F	3A 409B	2256	LDA LODFLAG ;Check the Load Flag
4E32	1F	2257	RAR ;Put it in the Carry
4E33	DC 4E28	2258	CC DCHK1 ;If this is a load command and a delimiter
		2259	;other than Dot is used, it's the end.
		2260	;After setting the DFLAG we will come back
		2261	;and set the AFLAG and no more File name or
		2262	;Extension will be accepted.
4E36	0A	2263	ACHK1: LDAX B ;Else, set the Address Flag to indicate
4E37	F6 02	2264	ORI AFLAG ;termination of the Address Field.
4E39	02	2265	STAX B ;Store it.
4E3A	C9	2266	RET ;Leave.
		2267	
		2268	
4E3B	01 7213	2269	DOT: LXI B,FLAGS ;Point to the Flags
4E3E	0A	2270	LDAX B ;Get them
4E3F	E6 42	2271	ANI AAFLAG ;Save the A Active and A Flags
4E41	FE 40	2272	CPI AACTIV ;Only A Active should be set
4E43	C4 5D1E	2273	CNZ SYNERR ;If not equal, it's an error
4E46	CD 4E36	2274	CALL ACHK1 ;Go set the A flag
4E49	C3 4D33	2275	JMP PARSE1 ;Go look for the next character
		2276	;
		2277	;.....
		2278	

Error	Addr	Code	Seq	Source statement
			2279	*****
			2280	;
			2281	;
			2282	;
			2283	;
			2284	;
4E4C	CD 4E20		2285	SPCFLW: CALL DLMCHK ;Go check the delimiter address/data field
			2286	;
4E4F	0A		2287	SPCF1: LDAX B ;Get a fresh copy of Flags
4E50	F6 01		2288	ORI SPCFLG ;Set the Space Flag
4E52	02		2289	STAX B ;Store it.
4E53	C3 4D33		2290	JMP PARSE1 ;Go do some more parsing.
			2291	;
			2292	*****
			2293	;
			2294	*****
			2295	;
			2296	;
			2297	;
			2298	;
			2299	;
			2300	;
4E56	CD 4E20		2301	SLASH: CALL DLMCHK ;Go do a Delimiter Check
4E59	0A		2302	LDAX B ;Get Flags
4E5A	E6 FE		2303	ANI 0FEH ;Clear the Space Flag
4E5C	02		2304	STAX B ;Store it.
4E5D	CD 4FC5		2305	CALL GETNXT ;Get the next character
4E60	21 7138		2306	LXI H,COUNT2 ;HL gets the Compare Table pointer.
4E63	11 7145		2307	LXI D,ADRES2 ;DE gets the Address Table pointer.
4E66	CD 4CFA		2308	CALL DECODE ;Go decode the switch.
4E69	CD 5D1E		2309	CALL SYNERR ;If we return it was an illegal switch.
			2310	;
			2311	*****
			2312	;
			2313	*****
			2314	;
			2315	;
			2316	;
			2317	;
4E6C	3E 01		2318	PHYS: MVI A,01H ;Physical
4E6E	C3 4E8C		2319	JMP COMMON ;
4E71	3E 02		2320	VIRTUAL: MVI A,02H ;Virtual
4E73	C3 4E8C		2321	JMP COMMON ;
4E76	3E 03		2322	GPR: MVI A,03H ;GPR
4E78	C3 4E8C		2323	JMP COMMON ;
4E7B	3E 04		2324	MREG: MVI A,04H ;Machine Dependent Registers
4E7D	C3 4E8C		2325	JMP COMMON ;
4E80	3E 05		2326	IREG: MVI A,05H ;Internal Registers
4E82	C3 4E8C		2327	JMP COMMON ;
4E85	3E 06		2328	MICRO: MVI A,06H ;8085 Ram

Error	Addr	Code	Seq	Source statement	
4E87	C3	4E8C	2329	JMP	COMMON
4E8A	3E	07	2330	WCS: MVI	A,07H
4E8C	32	71B3	2331	COMMON: STA	NEWAS
4E8F	C3	4D33	2332	JMP	PARSE1
			2333	;	
			2334	;.....	
			2335	;	
			2336	;.....	
			2337	;	
4E92	3E	40	2338	BYTE: MVI	A,040H
4E94	C3	4E9E	2339	JMP	COMMN1
4E97	3E	80	2340	WORD: MVI	A,080H
4E99	C3	4E9E	2341	JMP	COMMN1
4E9C	3E	30	2342	LONG: MVI	A,030H
4E9E	32	71B2	2343	COMMN1: STA	NEWDL
4EA1	C3	4D33	2344	JMP	PARSE1
			2345	;	
			2346	;.....	
			2347	;	
			2348	;	
			2349	;	
			2350	;.....	
			2351	;	
4EA4	3A	7215	2352	NEXTAD: LDA	FLAGS2
4EA7	A7		2353	ANA	A
4EA8	C4	5D1E	2354	CNZ	SYNERR
4EAB	CD	4FCS	2355	CALL	GETNXT
4EAE	FE	3A	2356	CPI	'.'
4EB0	C4	5D1E	2357	CNZ	SYNERR
4EB3	21	7208	2358	LXI	H,NBUFFER
4EB6	CD	6AFA	2359	CALL	ZROBUF
4EB9	CD	4FCS	2360	NEXTA1: CALL	GETNXT
4EBC	FE	2F	2361	CPI	'/'
4EBE	CA	4EDC	2362	JZ	NEXTA2
4EC1	FE	0D	2363	CPI	CR
4EC3	CA	4EDC	2364	JZ	NEXTA2
4EC6	FE	20	2365	CPI	' '
4EC8	CA	4EDC	2366	JZ	NEXTA2
4ECB	21	7215	2367	LXI	H,FLAGS2
4ECE	36	02	2368	MVI	M,2
4ED0	CD	4DF4	2369	CALL	CNVERT
4ED3	21	7208	2370	LXI	H,NBUFFER
4ED6	CD	4E0D	2371	CALL	SHIFT
4ED9	C3	4EB9	2372	JMP	NEXTA1
			2373	;	
4EDC	21	7215	2374	NEXTA2: LXI	H,FLAGS2
4EDF	7E		2375	NEXTA3: MOV	A,M
4EE0	E6	02	2376	ANI	2
4EE2	CC	5D1E	2377	CZ	SYNERR
4EE5	36	01	2378	MVI	M,1

Error	Addr	Code	Seq	Source statement	
4EE7	2A 40CA	2379		LHLD BUFADR	;Get the address
4EEA	2B	2380		DCX H	;Back it up
4EEB	22 40CA	2381		SHLD BUFADR	;Save it
4EEE	C3 4D33	2382		JMP PARSE1	;Go back
		2383			
4EF1	3A 7216	2384	STRTAD:	LDA FLAGS3	;Get this flags byte
4EF4	A7	2385		ANA A	;Is the S Flag set?
4EF5	C4 5D1E	2386		CNZ SYNERR	;Error if set
4EF8	CD 4FC5	2387		CALL GETNXT	;Get the next character
4EFB	FE 3A	2388		CPI ':'	;Check for correct syntax
4efd	C4 5D1E	2389		CNZ SYNERR	;Error if not
4F00	21 4111	2390		LXI H,SBUFFR	;Point to this buffer
4F03	CD 6AFA	2391		CALL ZROBUF	;Go zero the buffer
4F06	CD 4FC5	2392	STRTA1:	CALL GETNXT	;Get the next character
4F09	FE 40	2393		CPI ','	;Look for sign
4F0B	CA 4F2E	2394		JZ STRTA2	;Jump if equal
4F0E	FE 20	2395		CPI SPACE	;Is it a space?
4F10	CA 4F45	2396		JZ STRTA3	;Exit if it is
4F13	FE 2F	2397		CPI '/'	;Else, see if it is a slash
4F15	CA 4F45	2398		JZ STRTA3	;Exit if it is
4F18	FE 0D	2399		CPI CR	;Else, check for Carriage Return.
4F1A	CA 4F45	2400		JZ STRTA3	;Exit if it is
4F1D	21 7216	2401		LXI H,FLAGS3	;Point to this flags byte
4F20	36 02	2402		MVI M,2	;Set the S Active flag
4F22	CD 4DF4	2403		CALL CNVERT	;Else, go change from ASCII to Hex.
4F25	21 4111	2404		LXI H,SBUFFR	;Point to this buffer
4F28	CD 4E0D	2405		CALL SHIFT	;Go shift in the digit.
4F2B	C3 4F06	2406		JMP STRTA1	;Keep looping and looking
		2407			
4F2E	21 7216	2408	STRTA2:	LXI H,FLAGS3	;Point to the S Active and S flags
4F31	7E	2409		MOV A,M	;Get them
4F32	E6 02	2410		ANI 2	;Is the active flag set
4F34	C4 5D1E	2411		CNZ SYNERR	;Error if it is
4F37	36 03	2412		MVI M,3	;Set both flags
4F39	21 4111	2413		LXI H,SBUFFR	;Point to this dest
4F3C	11 71BC	2414		LXI D,EDDATA	;Point to this source
4F3F	CD 00F9	2415		CALL COPY	;Copy if over
4F42	C3 4D33	2416		JMP PARSE1	;Go back up the pointer
		2417			
4F45	21 7216	2418	STRTA3:	LXI H,FLAGS3	;Point to this byte
4F48	C3 4EDF	2419		JMP NEXTA3	;Go share some code
		2420			
		2421		;	
		2422			
		2423			
		2424		;	
		2425		;	
		2426		;Next come the routines to handle the address mnemonics and the +, -,	
		2427		;*, and symbols.	
		2428		;	

Error Addr	Code	Seq	Source statement		
4F4B	CD 4FB2	2429	PXXX:	CALL	ADRTST
4F4E	CD 4FC5	2430		CALL	GETNXT
		2431			;Check the address situation
					;Go get the next character and update
					;the pointer.
4F51	FE 43	2432		CPI	'C'
4F53	CA 4F79	2433		JZ	PRGCNT
4F56	FE 53	2434		CPI	'S'
4F58	C4 5D1E	2435		CNZ	SYNERR
4F5B	7E	2436		MOV	A,M
4F5C	FE 4C	2437		CPI	'L'
4F5E	C2 4F65	2438		JNZ	NOTL
4F61	23	2439		INX	H
4F62	22 40CA	2440		SHLD	BUFADR
4F65	3E 00	2441	NOTL:	MVI	A,PSLADR
4F67	32 71FE	2442		STA	ABUFFR
4F6A	C3 4E7B	2443		JMP	MREG
		2444			;Let this code set up the Address Space
4F6D	CD 4FB2	2445	RN:	CALL	ADRTST
4F70	CD 4FC5	2446		CALL	GETNXT
4F73	CD 4DF4	2447		CALL	CNVERT
4F76	C3 4F8B	2448		JMP	SHARE
		2449			;Go share some code
4F79	3E 0F	2450	PRGCNT:	MVI	A,0FH
4F7B	C3 4F8B	2451		JMP	SHARE
		2452			;Go share some code
4F7E	CD 4FB2	2453	STKFLW:	CALL	ADRTST
4F81	CD 4FC5	2454		CALL	GETNXT
4F84	FE 50	2455		CPI	'P'
4F86	C4 5D1E	2456		CNZ	SYNERR
4F89	3E 0E	2457		MVI	A,0EH
4F8B	32 71FE	2458	SHARE:	STA	ABUFFR
4F8E	C3 4E76	2459		JMP	GPR
		2460			;Go share some code
		2461	;.....		
		2462			
		2463			
		2464	;.....		
		2465			
4F91	06 01	2466	PLUS:	MVI	B,01H
4F93	C3 4FA2	2467		JMP	SHAREX
		2468			;This is the Plus Flag
					;Go to the shared code.
4F96	06 02	2469	MINUS:	MVI	B,02H
4F98	C3 4FA2	2470		JMP	SHAREX
		2471			;This will be used to set the Minus Flag
					;Go to the shared code.
4F9B	06 80	2472	ATSIGN:	MVI	B,080H
4F9D	C3 4FA2	2473		JMP	SHAREX
		2474			;This will be used to set the Flag.
					;
4FA0	06 04	2475	ASTRSK:	MVI	B,04H
		2476			;This will be used to set the Asterisk Flag
4FA2	21 7213	2477	SHAREX:	LXI	H,FLAGS
4FA5	CD 4FBA	2478		CALL	ADRTST
					;Point to the Flags byte
					;Check the address situation

Error	Addr	Code	Seq	Source statement
	4FAB	E6 BF	2479	ANI 0BFH ;Clear the A Active bit
	4FAA	F6 01	2480	ORI 1 ;Set the Space Flag
	4FAC	77	2481	MOV M,A ;and store the Flags
	4FAD	23	2482	INX H ;Point to the FLAGS1 byte
	4FAE	70	2483	MOV M,B ;Store the Flags
	4FAF	C3 4D33	2484	JMP PARSE1 ;Go back to the parser
			2485	;
			2486	;
			2487	;
			2488	;
			2489	;
			2490	;
			2491	ADRTST
			2492	;
			2493	;ADRTST checks the Space Flag and the Address Flag. An address field must
			2494	be preceded by a least one space. Also the parser must not have already
			2495	found an address field in the input string. It is not necessary to
			2496	check the A Active Flag because the Space Flag will be clear if the
			2497	parser is in the middle of parsing an address field.
			2498	;
	4FB2	21 7213	2499	ADRTST: LXI H,FLAGS ;Set up the pointer
	4FB5	7E	2500	MOV A,M ;Get the Flags
	4FB6	1F	2501	RAR ;Check the Space Flag
	4FB7	D4 5D1E	2502	CNC SYNERR ;If not set, error
	4FBA	7E	2503	ADRTST: MOV A,M ;Else, get a fresh copy of the Flags
	4FBB	E6 42	2504	ANI AAFLAG ;Check the A and Active Flags
	4FBD	C4 5D1E	2505	CNZ SYNERR ;Neither should be set
	4FC0	7E	2506	MOV A,M ;Get another copy
	4FC1	F6 42	2507	ORI AAFLAG ;Set both flags.
	4FC3	77	2508	MOV M,A ;Store it
	4FC4	C9	2509	RET ;Leave
			2510	;
			2511	;
			2512	;
			2513	;
			2514	;
			2515	;
			2516	GETNXT
			2517	;
			2518	;GETNXT loads the next character into A and updates the value of the
			2519	Character String Buffer address.
			2520	;
	4FC5	2A 40CA	2521	GETNXT: LHLD BUFADR ;Load the pointer
	4FC8	7E	2522	MOV A,M ;Get the next character
	4FC9	23	2523	INX H ;Bump the pointer
	4FCA	22 40CA	2524	SHLD BUFADR ;Save the new value
	4FCD	C9	2525	RET ;Go back
			2526	;
			2527	;
			2528	;

Error Addr	Code	Seq	Source statement
		2529	
		2530	;*****
		2531	;
		2532	; SEMI,CRFLOW
		2533	;
4FCE	CD 4FC5	2534	SEMI: CALL GETNXT ;Get the next character
4FD1	FE 0D	2535	CPI CR ;Look for a carriage return
4FD3	C2 4FCE	2536	JNZ SEMI ;Keep trying
4FD6	LD 4E20	2537	CRFLOW: CALL DLMCHK ;Do a Delimiter check
4FD9	0A	2538	LDAX B ;Get Flags
4FDA	E6 FE	2539	ANI 0FEH ;Clear the Space flag
4FDC	02	2540	STAX B ;Store it
4FDD	C9	2541	RET ;Leave
		2542	;
		2543	;*****
		2544	;
		2545	;*****
		2546	;
		2547	; DDECOD
		2548	;
4FDE	2A 40CA	2549	DDECOD: LHLD BUFADR ;Get the address of the character
4FE1	7E	2550	MOV A,M ;Get the character
4FE2	FE 49	2551	CPI 'I' ;Is it I?
4FE4	C2 5034	2552	JNZ DPOSIT ;Must be a deposit if not DIR
4FE7	23	2553	INX H ;Bump the pointer
4FE8	7E	2554	MOV A,M ;Get the next character
4FE9	FE 52	2555	CPI 'R' ;Is it an R?
4FEB	C2 5034	2556	JNZ DPOSIT ;Go to Deposit if not
4FEC	23	2557	INX H ;Bump the pointer
4FEF	3A 4150	2558	DIRECT: LDA DEFDRV ;Get the Default Drive number
4FF2	32 4091	2559	STA UNIT ;Prime the unit number
4FF5	CD 4D23	2560	CALL SPSTRP ;Go to Space strip
4FF8	FE 0D	2561	CPI CR ;Check for carriage return
4FFA	CA 5026	2562	JZ DIR1 ;Jump if it is
4FFD	FE 44	2563	CPI 'D' ;Look for a D
4FFF	C4 5D1E	2564	CNZ SYNERR ;Error if not
5002	23	2565	INX H ;Bump the pointer
5003	7E	2566	MOV A,M ;Get the next character
5004	FE 44	2567	CPI 'D' ;Is it a D
5006	C4 5D1E	2568	CNZ SYNERR ;Error if not
5009	23	2569	INX H ;Bump the pointer
500A	7E	2570	MOV A,M ;Get the next character
5009	CD 4DF4	2571	CALL CNVERT ;Go convert it to hex
500E	FE 02	2572	CPI 02 ;See if 0 or 1
5010	D4 5D1E	2573	CNZ SYNERR ;Error if not
5013	32 4091	2574	STA UNIT ;Set the unit number
5016	23	2575	INX H ;Bump the pointer
5017	7E	2576	MOV A,M ;Get the : (we hope)
5018	FE 3A	2577	CPI ':' ;Look for :
501A	C4 5D1E	2578	CNZ SYNERR ;Leave if not

Error Addr	Code	Seq	Source statement
501D	23	2579	INX H ;Bump the pointer again
501E	CD 4D23	2580	CALL SPSTRP ;Strip the spaces
5021	FE 0D	2581	CPI CR ;Look for carriage return
5023	C4 5D1E	2582	CNZ SYNERR ;Leave if not
5026	CD 46CB	2583	DIR1: CALL LSTPRP ;Go to List Prep
5029	3A 4116	2584	LDA SUCCES ;Check the success code
502C	A7	2585	ANA A ;Is it 0?
502D	C4 5D2D	2586	CNZ TAPERR ;Error if not
5030	32 409C	2587	STA DIRFLG ;Zero this flag
5033	C9	2588	RET ;Leave
		2589	;
		2590	*****
		2591	
		2592	
		2593	*****
		2594	;
		2595	;
		2596	DPOSIT
5034	CD 4D2B	2597	DPOSIT: CALL PARSE ;Go parse the input string
5037	3A 7213	2598	LDA FLAGS ;Get the Flags byte
503A	E6 04	2599	ANI DFLAG ;Deposit always requires data
503C	CC 5D1E	2600	CZ SYNERR ;If no data, error
503F	21 71B6	2601	LXI H,EDPACK ;Point to the first byte of the E/D Packet
5042	36 01	2602	MVI M,DEPCOD ;Put the Deposit Code there
5044	3A 7214	2603	LDA FLAG1 ;Check the Flag
5047	17	2604	RAL ;See if it is set
5048	DC 52DF	2605	CC ATFLOW ;Service it if it is.
504B	11 7202	2606	LXI D,DBUFFR ;Pointer to the data source.
504E	21 71BC	2607	LXI H,EDDATA ;Pointer to the data destination
5051	CD 00F9	2608	CALL COPY ;Move four byte from source to destination
5054	3A 7213	2609	LDA FLAGS ;See if there was an explicit new address
5057	E6 40	2610	ANI AACTIV ;by testing the Active bit
5059	CA 5065	2611	JZ DPOS1 ;If no explicit address, jump
505C	11 71FE	2612	LXI D,ABUFFR ;Else, point to the new address
505F	21 71B8	2613	LXI H,EDADDR ;Point to the destination
5062	CD 00F9	2614	CALL COPY ;Move source to destination
5065	CD 52AE	2615	DPOS1: CALL DECODX ;Go figure out the qual'ifiers
5068	3A 71B5	2616	DPOS2: LDA ADRSPC ;Get the Address Space byte
506B	FE 06	2617	CPI 6 ;Check for P,V,G,M or I space
506D	DC 508F	2618	CC MEMDEP ;Go to service routine if one of them
5070	FE 06	2619	CPI 6 ;Is this a 8085 Ram deposit?
5072	CC 50A5	2620	CZ MICDEP ;Go service it if it is
5075	FE 07	2621	CPI 7 ;Is this a WCS deposit?
5077	CC 50C3	2622	CZ WCSDEP ;Service it if it is
507A	E7	2623	RST 4 ;Go look for control characters
507B	3A 7217	2624	LDA RFLAG ;Get the R Flag
507E	1F	2625	RAR ;Is it set?
507F	D2 5089	2626	JNC NPREP ;Jump if not set
5082	D3 3D	2627	OUT SETLIT ;Generate a sync pulse
5084	D3 3C	2628	OUT CLRLIT ;using the Run light

Error	Addr	Code	Seq	Source statement	
5086	C3	5068	2629	JMP	DPOS2
5089	CD	56BE	2630	NPREP: CALL	NTEST
508C	C3	5065	2631	JMP	DPOS1
			2632		
			2633		
508F	FE	04	2634	MEMDEP: CPI	04H
5091	CC	53BA	2635	CZ	MCHECK
5094	CD	5373	2636	CALL	MAKMOD
5097	CD	537D	2637	CALL	ITEST
509A	CD	514B	2638	CALL	SNDMSX
509D	CD	53A2	2639	CALL	ICHECK
50A0	CD	5186	2640	CALL	RCVMSX
50A3	AF		2641	XRA	A
50A4	C9		2642	RET	
			2643		
			2644		
50A5	3A	71B9	2645	MICDEP: LDA	EDADDR+1
50A8	FE	FF	2646	CPI	0FFH
50AA	CA	50B6	2647	JZ	DEPIO
50AD	2A	71B8	2648	LHLD	EDADDR
50B0	3A	71BC	2649	LDA	EDDATA
50B3	77		2650	MOV	M,A
50B4	AF		2651	XRA	A
50B5	C9		2652	RET	
			2653		
50B6	3A	71B8	2654	DEPIO: LDA	EDADDR
50B9	32	50C0	2655	STA	IOARG1+1
50BC	3A	71BC	2656	LDA	EDDATA
50BF	D3	84	2657	IOARG1: OUT	UPC14
50C1	AF		2658	XRA	A
50C2	C9		2659	RET	
			2660		
			2661		
50C3	D3	20	2662	WCSDEP: OUT	CLRCLK
50C5	D3	A7	2663	OUT	DISMEM
50C7	11	71B8	2664	LXI	D,EDADDR
50CA	21	41AA	2665	LXI	H,SHAP
50CD	22	4176	2666	SHLD	POINTR
50D0	CD	00F9	2667	CALL	COPY
50D3	CD	49D4	2668	CALL	LDUPC1
50D6	CD	4993	2669	CALL	WRTNOP
50D9	21	71BC	2670	LXI	H,EDDATA
50DC	CD	5D05	2671	CALL	WRTWCS
50DF	CD	49A7	2672	CALL	LDCSR1
50E2	CD	49D4	2673	CALL	LDUPC1
50E5	D3	A6	2674	OUT	ENBMEM
50E7	3A	4196	2675	LDA	CLKFLG
50EA	1F		2676	RAR	
50EB	D2	50F0	2677	JNC	WCSDE1
50EE	D3	21	2678	OUT	SETCLK

;and loop
 ;Else, go see if the N Flag is set
 ;Go back and figure out the switches.

;Check for M space
 ;Go to MCHECK if it is
 ;Go build the EDMOD byte
 ;Go see about changing the modifier byte
 ;Send the Data to the Base machine
 ;Go check for I space Console address
 ;Go get the message
 ;Fake out the rest of the inline compares
 ;Go back

;Check 8085 I/O address
 ;Check it
 ;Jump if I/O specified
 ;Get the address
 ;Get the Data
 ;Do the Deposit
 ;Fake out the rest of the inline compares
 ;Leave

;Get the low byte of the address
 ;Change I stream
 ;Get the data
 ;Dummy OUT address
 ;Fake out the rest of the inline compares
 ;Leave

;Stop the CPU clocks
 ;Turn off memory
 ;Point to the address
 ;Point to the Swapping buffer
 ;Save the Address
 ;Copy it
 ;Go load the UPC
 ;NOP the CSR
 ;Point to the data
 ;Go write it
 ;Go restore the old CSR
 ;Restore the old UPC
 ;Turn on memory
 ;Get the Clock Flag
 ;Is it set?
 ;If it isn't set leave the clocks off
 ;Otherwise, set the clock

Error	Addr	Code	Seq	Source statement
50F0	AF		2679	WCSDE1: XRA A ;Fake out the rest of the inline compares
50F1	C9		2680	RET ;Leave
			2681	;
			2682	;*****
			2683	
			2684	
			2685	;*****
			2686	;
			2687	; EXAMIN
			2688	;
50F2	CD 4D2B		2689	EXAMIN: CALL PARSE ;Go parse the input string
50F5	3A 7213		2690	LDA FLAGS ;Going to check for data
50F8	E6 04		2691	ANI DFLAG ;Is there any?
50FA	C4 5D1E		2692	CNZ SYNERR ;If set, error because there shouldn't be
50FD	AF		2693	XRA A ;Generate 0
50FE	32 71B6		2694	STA EDPACK ;This is the Examine Code
5101	3A 7214		2695	LDA FLAGS1 ;Check for Flag
5104	17		2696	RAL ;See if it is set
5105	DC 52DF		2697	CC ATFLOW ;If it is, go service it
5108	21 71BC		2698	LXI H,EDDATA ;Point to the Examin/Deposit Data area
510B	CD 6AFA		2699	CALL ZROBUF ;Go zero it
510E	3A 7213		2700	LDA FLAGS ;Check for an explicit address
5111	E6 40		2701	ANI AACTIV ;Is the Address Active Flag set
5113	CA 511F		2702	JZ EXAMI1 ;If not, jump
5116	11 71FE		2703	LXI D,ABUFFR ;Point to the Address Buffer
5119	21 71B8		2704	LXI H,EDADDR ;Point to the Packet address area
511C	CD 00F9		2705	CALL COPY ;Go transfer it
511F	CD 52AE		2706	EXAMI1: CALL DECODX ;Go figure out the qualifiers
5122	3A 71B5		2707	EXAMI2: LDA ADRSPC ;Get the Address Space byte
5125	FE 06		2708	CPI 6 ;Check for P,V,G,M or I space
5127	DC 51C1		2709	CC MEMEXM ;Go service them if ADRSPC<4
512A	FE 06		2710	CPI 6 ;Is it 8085 Ram
512C	CC 51E0		2711	CZ MICEXM ;Call if equal
512F	FE 07		2712	CPI 7 ;Is it WCS?
5131	CC 51FE		2713	CZ WCSEXM ;Call if equal
5134	CD 523A		2714	CALL EXTTYPE ;Print the Address and Data
5137	3A 7217		2715	LDA RFLAG ;Get the R Flag
513A	1F		2716	RAR ;Is it set?
513B	D2 5145		2717	JNC NPREP1 ;Jump if no R Flag
513E	D3 3D		2718	OUT SETLIT ;Use the Run light to generate a sync pulse
5140	D3 3C		2719	OUT CLRLIT ;
5142	C3 5122		2720	JMP EXAMI2 ;Then jump back
			2721	
5145	CD 56BE		2722	NPREP1: CALL NTEST ;Go to the common NTEST subroutine
5148	C3 511F		2723	JMP EXAMI1 ;If we return, it is because the N Flag is set
			2724	;and the count is not 0
514B	21 7 B6		2725	SNDMSX: LXI H,EDPACK ;Point to the packet to be sent
514E	06 0C		2726	SNDMSG: MVI B,05 ;Count for the number of two byte transfers
5150	3A 4196		2727	SNDMS0: LDA CLKFLG ;Get the clock flag
5153	A7		2728	ANA A ;Check it

Error	Addr	Code	Seq	Source statement	
5154	CC	5D70	2729	CZ	CLKERR
5157	0E	00	2730	SNDMS1: MVI	C,0
5159	7E		2731	MOV	A,M
515A	23		2732	INX	H
515B	D3	CC	2733	OUT	WRITE
515D	D3	AA	2734	OUT	SETATN
515F	DB	82	2735	SNDMS2: IN	CPUACK
5161	1F		2736	RAR	
5162	DA	516C	2737	JC	SNDMS3
5165	0D		2738	DCR	C
5166	C2	515F	2739	JNZ	SNDMS2
5169	CD	5D5E	2740	CALL	COMERR
516C	7E		2741	SNDMS3: MOV	A,M
516D	23		2742	INX	H
516E	D3	CC	2743	OUT	WRITE
5170	D3	AB	2744	OUT	CLRATN
5172	0E	00	2745	MVI	C,0
5174	DB	82	2746	SNDMS4: IN	CPUACK
5176	1F		2747	RAR	
5177	D2	5181	2748	JNC	SNDMS5
517A	0D		2749	DCR	C
517B	C2	5174	2750	JNZ	SNDMS4
517E	CD	5D5E	2751	CALL	COMERR
5181	05		2752	SNDMS5: DCR	B
5182	C2	5157	2753	JNZ	SNDMS1
5185	C9		2754	RFT	
			2755		
5186	21	71C0	2756	RCVMSX: LXI	H,RCVBUF
5189	E5		2757	RCVMSG: PUSH	H
518A	06	05	2758	MVI	B,05H
518C	0E	00	2759	RCVMS5: MVI	C,0
518E	DB	83	2760	RCVMS1: IN	CPATTN
5190	1F		2761	RAR	
5191	DA	519B	2762	JC	RCVMS2
5194	0D		2763	DCR	C
5195	C2	518E	2764	JNZ	RCVMS1
5198	CD	5D5E	2765	CALL	COMERR
519B	DB	80	2766	RCVMS2: IN	READ
519D	D3	AB	2767	OUT	SETACK
519F	77		2768	MOV	M,A
51A0	23		2769	INX	H
51A1	0E	00	2770	MVI	C,0
51A3	DB	83	2771	RCVMS3: IN	CPATTN
51A5	1F		2772	RAR	
51A6	D2	51B0	2773	JNC	RCVMS4
51A9	0D		2774	DCR	C
51AA	C2	51A3	2775	JNZ	RCVMS3
51AD	CD	5D5E	2776	CALL	COMERR
51B0	DB	80	2777	RCVMS4: IN	READ
51B2	D3	A9	2778	OUT	CLRACK

Error Addr	Code	Seq	Source statement	
51B4	77	2779	MOV M,A	;Store the Data
51B5	23	2780	INX H	;Bump the pointer
51B6	05	2781	DCR B	;Decrement the count
51B7	C2 51B8	2782	JNZ RCVMS5	;If not zero, do it again
51BA	E1	2783	POP H	;Get the address of the first byte of storage
51BB	7E	2784	MOV A,M	;Get the success code byte
51BC	A7	2785	ANA A	;Is it 0?
51BD	C8	2786	RZ	;Leave if it is
51BE	CD 5D46	2787	CALL MEMERR	;Error if not
		2788		
		2789		
51C1	FE 04	2790	MEMEXM: CPI 4	;Check for M space
51C3	CC 53BA	2791	CZ MCHECK	;Leave if it is
51C6	CD 5373	2792	CALL MAKMOD	;Go build the EDMOD byte
51C9	CD 537D	2793	CALL ITEST	;Go see if we need to alter the modifier byte
51CC	CD 514B	2794	CALL SNDMSX	;Tell the CPU what we want
51CF	CD 53A2	2795	CALL ICHECK	;Go see if this is I space Console address
51D2	CD 5186	2796	CALL RCVMSX	;Go get the reply
51D5	21 71BC	2797	LXI H,EDDATA	;Destination
51D8	11 71C6	2798	LXI D,RCVDAT	;Source
51DB	CD 00F9	2799	CALL COPY	;Go move the data
51DE	AF	2800	XRA A	;Make sure the rest of the compares will fail
51DF	C9	2801	RET	;Go back
		2802		
		2803		
51E0	3A 71B9	2804	MICEXM: LDA EDADDR+1	;Get the second byte
51E3	FE FF	2805	CPI 0FFH	;See if the address specified is I/O
51E5	CA 51F1	2806	JZ EXMIO	;Go examine the I/O area
51E8	2A 71B8	2807	LHLD EDADDR	;Get the address
51EB	7E	2808	MOV A,M	;Get the data
51EC	32 71BC	2809	STA EDDATA	;Put it away
51EF	AF	2810	XRA A	;Make sure the next inline compare fails
51F0	C9	2811	RET	;Go back
		2812		
51F1	3A 71B8	2813	EXMIO: LDA EDADDR	;Get the address
51F4	32 51F8	2814	STA IOARG2+1	;Modify I stream
51F7	DB 84	2815	IOARG2: IN UPC14	;Dummy address
51F9	32 71BC	2816	STA EDDATA	;Save the data
51FC	AF	2817	XRA A	;Force miscompare for next inline compare
51FD	C9	2818	RET	;Leave
		2819		
		2820		
51FE	2A 71B8	2821	WCSEXM: LHLD EDADDR	;Get the Address to be examined
5201	D3 A7	2822	OUT DISMEM	;Turn off memory
5203	CD 521F	2823	CALL WCSEX1	;Call the subroutine part of this
5206	D3 A6	2824	OUT ENBMEM	;Turn memory back on
5208	2A 41A7	2825	LHLD NOPBUF	;Get the low two bytes
520B	22 71BC	2826	SHLD EDDATA	;Store them
520E	3A 41A9	2827	LDA NOPBUF+2	;Get the High byte
5211	32 71BE	2828	STA EDDATA+2	;Save it also

Error Addr	Code	Seq	Source statement	
5214	3A 4196	2829	LDA CLKFLG	;Get the Clock Flag
5217	1F	2830	RAR	;Is it set?
5218	D2 521D	2831	JNC WCSEX2	;Leave if not set
521B	D3 21	2832	OUT SETCLK	;Else set the clocks
521D	AF	2833	WCSEX2: XRA A	;Fake any more inline compares
521E	C9	2834	RET	;Leave
		2835		
521F	22 41AA	2836	WCSEX1: SHLD SWAP	;Put the address into the Swap buffer
5222	D3 20	2837	OUT CLRCLK	;Stop the CPU Clock
5224	21 41AA	2838	LXI H SWAP	;Point to the Swap buffer
5227	CD 49D1	2839	CALL LDUPC	;Go load it into the UPC
522A	CD 4993	2840	CALL WRTNOP	;Go write a NOP
522D	D3 A1	2841	OUT DISPAR	;Turn off parity
522F	D3 23	2842	OUT SETSS	;Single Step the machine to get the contents
5231	D3 22	2843	OUT CLRSS	;of the WCS location specified into the CSR
5233	CD 49A7	2844	CALL LDCSR1	;Put the WCS data into the NOPBUF and
		2845		;the old CSR back into the CSR
5236	CD 49D4	2846	CALL LDUPC1	;Put the old UPC back.
5239	C9	2847	RET	;Leave
		2848		
		2849		
		2850		
		2851		
		2852		
		2853		
523A	21 7450	2854	EXTYPE: LXI H,CODTAB	;Get the base address of this table into HL
523D	3A 71B5	2855	LDA ADRSPC	;Get the Address Space byte
5240	4F	2856	MOV C,A	;Put it in C
5241	06 00	2857	MVI B,0	;Clear B
5243	09	2858	DAD B	;Add it to HL
5244	7E	2859	MOV A,M	;Index into the Code Table
5245	32 7225	2860	STA OUTLET	;This is the ASCII letter of the Address Space
5248	06 04	2861	MVI B,04	;Count for converting Hex to ASCII
524A	21 71C9	2862	LXI H,RCVDAT+3	;Point to the high byte of the data
524D	79	2863	MOV A,C	;Retrieve the Address Space
524E	FE 02	2864	CPI 02	;Is it virtual?
5250	CA 5256	2865	JZ SKP11	;Jump if it is
5253	21 71BF	2866	LXI H,EDDATA+3	;Point here if it isn't
5256	11 7230	2867	SKP11: LXI D,OUTDAT	;Point to the Output Data Buffer
5259	CD 5DBD	2868	CALL HEXASC	;Call the Hex to ASCII routine
525C	06 04	2869	MVI B,04	;Prime the count again
525E	11 7227	2870	LXI D,OUTADR	;Point to the Output address buffer
5261	CD 5DBD	2871	CALL HEXASC	;Go convert the Address from Hex to ASCII
5264	CD 526C	2872	CALL SUPRES	;Go suppress leading zeros as necessary
5267	21 7224	2873	LXI H,OUTBUF	;Point to the Output Buffer
526A	F7	2874	RST 6	;Go print the Output Buffer
526B	C9	2875	RET	;Leave
		2876		
		2877		
		2878		

Error Addr	Code	Seq	Source statement
526C	3A 71B5	2879	SUPRES: LDA ADRSPC ;Get the Address Space
526F	FE 04	2880	CPI 04 ;Check for P,V or G Spaces
5271	D2 5289	2881	JNC NOTPVG ;Jump if not
5274	3A 71B4	2882	LDA DATLEN ;Get the Data Length
5277	FE 40	2883	CPI 040H ;Is it Byte?
5279	C2 5281	2884	JNZ NOTBYT ;Jump if not
527C	06 06	2885	SUPRS6: MVI B,06 ;Let's clear the upper 6 nibbles
527E	C3 5293	2886	JMP MAKSPC ;Go put spaces there
5281	FE 80	2887	NOTBYT: CPI 080H ;Is it word?
5283	C0	2888	RNZ ;If longword, leave it alone
5284	06 04	2889	SUPRS4: MVI B,04 ;Count for MAKSPC
5286	C3 5293	2890	JMP MAKSPC ;Go put spaces in the upper word
5289	FE 06	2891	NOTPVG: CPI 06 ;Is it I or M?
528B	DA 529E	2892	JC UPCTST ;See if we are dealing with the UPC
528E	CA 527C	2893	JZ SUPRS6 ;Go suppress 6 digits if it is the 8085 RAM
5291	06 02	2894	MVI B,02 ;Suppress 2 digits because it is the WCS
		2895	
5293	21 7230	2896	MAKSPC: LXI H,OUTDAT ;Point to the data to be printed
5296	36 20	2897	MKSPC1: MVI M,020H ;Send a space to the output buffer
5298	23	2898	INX H ;Bump the pointer
5299	05	2899	DCR B ;Decrement the count
529A	C2 5296	2900	JNZ MKSPC1 ;Loop if not done
529D	C9	2901	RET ;Leave
		2902	
529E	FE 04	2903	UPCTST: CPI 04 ;See if it is M space
52A0	C0	2904	RNZ ;Leave if not
52A1	21 71B8	2905	LXI H,EDADDR ;Point to the low byte of the address
52A4	3E 01	2906	MVI A,01 ;Put a 1 in A
52A6	BE	2907	CMP M ;Is the low byte of the address a 1?
52A7	C0	2908	RNZ ;Leave if not the same
52A8	23	2909	INX H ;Bump the pointer
52A9	BE	2910	CMP M ;See if the next byte is a one
52AA	C8	2911	RZ ;Leave if it is
52AB	C3 5284	2912	JMP SUPRS4 ;Go suppress 4 characters
		2913	
		2914	
		2915	
		2916	
		2917	
		2918	;DECODX is used by Examine and Deposit to figure out what the
		2919	;switches are saying to do. Incrementing and decrementing the address
		2920	;and most of the preparation of the packet for transmission to the
		2921	;base machine are performed by routines called from DECODX.
		2922	
52AE	21 71B5	2923	DECODX: LXI H,ADRSPC ;Point the to the address space byte
52B1	01 71B3	2924	LXI B,NEWAS ;Point to the New Address space
52B4	0A	2925	LDAX B ;Get it
52B5	E6 0F	2926	ANI 0FH ;Check for new A.S.=DEFAULT
52B7	CA 52D5	2927	JZ DFAULT ;If no new A.S. Qualifier, go check old
52BA	77	2928	REENTR: MOV M,A ;Update the Address Space byte

Error	Addr	Code	Seq	Source statement
	52BB	FE 03	2929	CPI 03H ;Is A.S. less than or equal 03H?
	52BD	DA 52E8	2930	JC PHYVIR ;If less than, must be a physical or
			2931	;virtual reference. (The console doesn't
			2932	;care which.)
	52C0	CA 5355	2933	JZ GFLOW ;If equal, must be a GPR reference
	52C3	FE 04	2934	CPI 04H ;Else, check for M registers
	52C5	CA 536D	2935	JZ MFLOW ;If equal, go service it
	52C8	FE 05	2936	CPI 05H ;Else, check for I registers
	52CA	CA 5367	2937	JZ IFLOW ;If equal, go service it
	52CD	FE 06	2938	CPI 06H ;Else, check for 8085 RAM SPACE
	52CF	CA 5361	2939	JZ UFLOW ;If equal, jump
	52D2	C3 535B	2940	JMP CFLOW ;Else, go to the WCS Space flows.
			2941	
			2942	
			2943	
			2944	;DFAULT is entered when the new Qualifier equals 0. This means there is
			2945	;no new Address Space. The old Qualifier
			2946	;is checked and if it also equals 0 the code defaults to PHYVIR.
			2947	
	52D5	7E	2948	DFAULT: MOV A,M ;Get the Address Space byte
	52D6	E6 0F	2949	ANI 0FH ;See if anything there
	52D8	CA 52E8	2950	JZ PHYVIR ;If no old A.S., default to PHYVIR
	52DB	02	2951	STAX B ;If there is an old A.S., copy it to NEWAS
	52DC	C3 52BA	2952	JMP REENTR ;Now go decode it
			2953	
			2954	
			2955	; This subroutine is used to handle the address space. This address
			2956	;space indicates that the last data is to be used as the current address.
			2957	
			2958	
	52DF	21 71B8	2959	ATFLOW: LXI H,EDADDR ;Point to the Examine/Deposit Address buffer
	52E2	11 71BC	2960	LXI D,EDDATA ;Point to the Examine/Deposit Data buffer
	52E5	C3 00F9	2961	JMP COPY ;Go copy
			2962	
			2963	
			2964	
			2965	
	52E8	3A 7213	2966	PHYVIR: LDA FLAGS ;Get the Flags
	52EB	E6 02	2967	ANI AFLAG ;See if any Address was specified
	52ED	CC 5305	2968	CZ UPPREP ;If no new address specifier, assume increment
	52F0	3A 7214	2969	LDA FLAGS1 ;Else, get the +, -, Flags
	52F3	1F	2970	RAR ;Is the + Flag set?
	52F4	DC 5305	2971	CC UPPREP ;If it is, go see about the D.L.
	52F7	1F	2972	RAR ;Is the - Flag set?
	52F8	DC 5329	2973	CC DNPREP ;If it is, go see about the D.L.
	52FB	3A 71B2	2974	MERGE: LDA NEWDL ;Get the New Data Length
	52FE	E6 F0	2975	ANI 0F0H ;Is it the D.L. Default?
	5300	C8	2976	RZ ;Leave if new=default
	5301	32 71B4	2977	STA DATLEN ;Update Data Length byte if new is not default
	5304	C9	2978	RET ;Leave

Error Addr Code Seq Source statement

```

2979
2980
2981 ; .....
2982 ;
2983 ;       UPPREP,FOURUP,TWOUP,ONEUP
2984 ;
2985 ;       This code figures out how much to add to the address. It then points
2986 ; to the correct amount and jumps to a subroutine that will add it for us.
2987 ; The return at the end of that subroutine will send us back to the caller
2988 ;
5305 3A 71B4 2989 UPPREP: LDA    DATLEN    ;Get the old data length
5308 E6 F0    2990         ANI    0F0H    ;Check for default
530A CA 5317 2991         JZ     FOURUP   ;If it was default, go add four to the address
530D FE 80    2992         CPI    080H   ;Else, check for Word
530F CA 531D 2993         JZ     TWOUP    ;If it was Word, go add two to the address
5312 FE 40    2994         CPI    040H   ;Else, check for Byte
5314 CA 5323 2995         JZ     ONEUP     ;If it was Byte, go add one to the address
5317 01 718E 2996 FOURUP: LXI    B,POS4   ;Else, add four because it must be Long
531A C3 624E 2997         JMP    ADDADR   ;Go add the proper amount
2998
531D 01 718A 2999 TWOUP:  LXI    B,POS2   ;Add two
5320 C3 624E 3000         JMP    ADDADR   ;Go add it
3001
5323 01 7186 3002 ONEUP:  LXI    B,POS1   ;Add one
5326 C3 624E 3003         JMP    ADDADR   ;Go add it
3004
3005 ; .....
3006
3007
3008
3009 ; .....
3010 ;
3011 ;       DNPREP,FOURDN,TWODN,ONEDN
3012 ;
3013 ;       DNPREP figures out which constant to point to and jumps as neccessary
3014 ; to code that loads up a pointer to the correct constant. Then a jump is made
3015 ; to the ADDADR subroutine which will add the constant in and perform the return
3016 ; for us.
3017 ;
5329 3A 71B2 3018 DNPREP: LDA    NEWDL    ;Get the new Data Length byte
532C E6 F0    3019         ANI    0F0H    ;Check if default
532E C2 5339 3020         JNZ    DNPRED1  ;Jump if not default
5331 3A 71B4 3021         LDA    DATLEN   ;Get the old Data Length byte
5334 E6 F0    3022         ANI    0F0H    ;Is it 0?
5336 CA 5343 3023         J7     FOURDN   ;If it is, prepare to subtract 4
5339 FE 80    3024 DNPRED1: CPI    080H   ;Is it Word?
533B CA 5349 3025         JZ     TWODN    ;If it is, go subtract two from the address
533E FE 40    3026         CPI    040H   ;Else, check for Byte
5340 CA 534F 3027         JZ     ONEDN     ;If it is, go subtract one from the address
5343 01 71A2 3028 FOURDN: LXI    B,NEG4   ;Point to negative 4

```

Error	Addr	Code	Seq	Source statement	
	5346	C3 624E	3029	JMP	ADDADR ;Go add it
			3030		
	5349	01 719A	3031	TWODN: LXI	B,NEG2 ;Point to negative 2
	534C	C3 624E	3032	JMP	ADDADR ;Go add it in.
			3033		
	534F	01 7196	3034	ONEDN: LXI	B,NEG1 ;Point to negative 1
	5352	C3 624E	3035	JMP	ADDADR ;Go add it
			3036		
			3037		;.....
			3038		
			3039		
			3040		
	5355	01 7458	3041	GFLOW: LXI	B,GPRCON ;Address of the four byte mask to be anded
			3042		;with the contents of the address buffer.
	5358	C3 568A	3043	JMP	COMFIN ;
			3044		
			3045		
	535B	01 7464	3046	CFLOW: LXI	B,WCS CON ;Point to the WCS constant
	535E	C3 568A	3047	JMP	COMFIN ;Go to the Common routine
			3048		
			3049		
	5361	01 7468	3050	UFLOW: LXI	B,UCON ;Point to the 8085 constant
	5364	C3 568A	3051	JMP	COMFIN ;
			3052		
			3053		
	5367	01 745C	3054	IFLOW: LXI	B,I ;Point to the I constant
	536A	C3 568A	3055	JMP	COM N ;Go to the Common finish
			3056		
	536D	01 7460	3057	MFLOW: LXI	B,MCON ;Load the pointer to the M constant
	5370	C3 568A	3058	JMP	COMFIN ;Let Comfin do some of the work
			3059		
			3060		
	5373	21 71B4	3061	MAKMOD: LXI	H,DATLEN ;Point to the Data Length
	5376	7E	3062	MOV	A,M ;Get it
	5377	23	3063	INX	H ;Point to the Address Space
	5378	B6	3064	ORA	M ;OR it in
	5379	32 71B7	3065	STA	EDMOD ;Put it in the Examine/Deposit Mod byte
	537C	C9	3066	RET	;Leave
			3067		
			3068		;.....
			3069		
	537D	3A 71B5	3070	ITEST: LDA	ADRSPC ;Get the address space
	5380	FE 05	3071	CPI	5 ;See if I space is specified
	5382	C0	3072	RNZ	;Leave if not I space
	5383	21 7223	3073	LXI	H,IFLAG ;Point to the I flag
	5386	36 01	3074	MVI	M,1 ;Set to inhibit SETATN while in PM
	5388	21 71B7	3075	ITEST1: LXI	H,EDMOD ;Point to the EDPACK modifier byte
	538B	7E	3076	MOV	A,M ;Get it
	538C	E6 F7	3077	ANI	0F7H ;Clear the MSB of the low nibble
	538E	77	3078	MOV	M,A ;Send it back out

Error Addr	Code	Seq	Source statement
538F	23	3079	INX H ;Bump pointer to EDADDR
5390	7E	3080	MOV A,M ;Get the address
5391	FE 18	3081	CPI 18H ;Check for low limit of Console Domain I space
5393	D8	3082	RC ;Leave if below Console I space low address
5394	FE 24	3083	CPI 24H ;Check for high limit
5396	DA 539C	3084	JC ITEST2 ;Jump if at or below high limit
5399	FE 37	3085	CPI 37H ;Check the one address not in the range
539B	C0	3086	RNZ ;Leave with non-Console I space mod in EDMOD
539C	2B	3087	ITEST2: DCX H ;Point back to EDMOD
539D	7E	3088	MOV A,M ;Get the modifier byte
539E	F6 08	3089	ORI 8H ;Set the MSB of the low nibble
53A0	77	3090	MOV M,A ;Put it back
53A1	C9	3091	RET ;Leave
		3092	
		3093	
53A2	3A 71B7	3094	ICHECK: LDA EDMOD ;Get the modifier byte
53A5	E6 0F	3095	ANI 0FH ;Save the low nibble
53A7	FE 0D	3096	CPI 0DH ;See if Console domain I space is specified
53A9	C0	3097	RNZ ;Leave if not
53AA	06 00	3098	MVI B,0 ;Otherwise, set up a timeout count
53AC	04	3099	ICHECK1: INR B ;Bump the time out count
53AD	CC 5D5E	3100	CZ COMERR ;Time out error if = 0
53B0	DB 83	3101	IN CPATTN ;Get the the CPU attention bit
53B2	1F	3102	RAR ;Put it in the carry
53B3	D2 53AC	3103	JNC ICHECK1 ;Loop back if not set
53B6	CD 6558	3104	CALL CPSERV ;Go service the CPU
53B9	C9	3105	RET ;We should not return here but one level
		3106	;higher instead
		3107	;
		3108	;*****
		3109	
		3110	
		3111	;*****
		3112	;
53BA	E1	3113	MCHECK: POP H ;Put the return address in HL temporarily
53BB	3A 71B9	3114	LDA EDADDR+1 ;Get the second byte of the address
53BE	FE 01	3115	CPI 01 ;Is it a one?
53C0	CA 549E	3116	JZ EDLS ;Go to the code that handles LS
53C3	3A 71B8	3117	LDA EDADDR ;Get the low byte of the address
53C6	FE 20	3118	CPI 020H ;Look for a Q register access
53C8	CA 54FA	3119	JZ EDQ ;Jump if it is
53CB	FE 10	3120	CPI 010H ;Check for less than 10 hex
53CD	D2 5517	3121	JNC EDWR ;Should be a Working Register access
53D0	FE 01	3122	CPI UPCADR ;Is it the UPC?
53D2	CA 5481	3123	JZ UPCCHK ;If it is, go check for Examine or Deposit
53D5	FE 04	3124	CPI 04 ;Check for the address to flip uword parity
53D7	CA 543D	3125	JZ FLIP ;Go flip the MSB of the uword
53DA	FE 05	3126	CPI 05 ;Look for the Enable/Disable Parity address
53DC	CA 540E	3127	JZ EDPAR ;Jump if it is
53DF	FE 06	3128	CPI 06 ;Look for CPU CSR address

Error	Addr	Code	Seq	Source statement	
53E1	CA	53E5	3129	JZ	EDCSR
53E4	E9		3130	PCHL	
			3131		
53E5	D3	20	3132	EDCSR: OUT	CLRCCLK
53E7	D3	A7	3133	OUT	DISMEM
53E9	CD	4993	3134	CALL	WRTNOP
			3135		
53EC	2	41A7	3136	LXI	H,NOPBUF
53EF	11	71BC	3137		D,EDD*TA
53F2	06	03	3138		B,3
53F4	3A	71B6	3139	LDA	EDPACK
53F7	FE	00	3140	CPI	EXMCOD
53F9	C2	43FD	3141	JNZ	EDCSR1
53FC	EB		3142	XCHG	
53FD	CD	00FB	3143	EDCSR1: CALL	COPY1
5400	CC	49A7	3144	CALL	LDCSR1
5403	D3	A6	3145	OUT	ENBMEM
5405	3A	4196	3146	LDA	CLKFLG
5408	A7		3147	ANA	A
5409	C8		3148	RZ	
540A	D3	21	3149	OUT	SETCLK
540C	AF		3150	XRA	A
540D	C9		3151	RET	
			3152		
540E	3A	71B6	3153	EDPAR: LDA	EDPACK
5411	FE	00	3154	CPI	EXMCOD
5413	CA	5435	3155	JZ	EXMPAR
5416	3A	71BC	3156	LDA	EDDATA
5419	A7		3157	ANA	A
541A	CA	542A	3158	JZ	MFLOW1
541D	32	4197	3159	STA	PARFLG
5420	D3	A0	3160	OUT	ENBPAR
5422	20		3161	RIM	
5423	F6	08	3162	ORI	08H
5425	E6	0D	3163	ANI	0DH
5427	30		3164	SIM	
5428	AF		3165	XRA	A
5429	C9		3166	RET	
542A	32	4197	3167	MFLOW1: STA	PARFLG
542D	D3	A1	3168	OUT	DISPAR
542F	20		3169	RIM	
5430	F6	0A	3170	ORI	0AH
5432	30		3171	SIM	
5433	AF		3172	XRA	A
5434	C9		3173	RET	
			3174		
5435	3A	4197	3175	EXMPAR: LDA	PARFLG
5438	32	71BC	3176	STA	EDDATA
543B	AF		3177	XRA	A
543C	C9		3178	RET	

```

;Jump if it is
;Return if address wasn't one the 8085 handles

;Turn off the CPU clock
;Turn off Main Memory
;This will get the contents of the CSR into the
;NOPBUF
;Point to the NOP Buffer
;Point to the data
;Set up the proper count
;Find out if it is Examine or Deposit
;Try the Examine Code
;Jump if Deposit
;If Examine, change the direction of copying
;Change the contents of the NOPBUF if Deposit
;Go load it in
;Turn on Main Memory
;Check the clock flag
;Is it set?
;Leave if not
;Otherwise, turn it on
;Fake out the inline compares
;and then leave

;Get the packet header to check for E or D
;Which is it?
;Go examine the parity bit
;Get the data
;See if it is zero
;Jump if it is
;Store the parity flag
;Turn on Halt on Parity
;Read in the masks
;Set the MSE
;Clear the RST 6.5 mask
;Write the masks
;Fake out the inline compares
;Leave
;Store the parity flag
;Turn off halt on parity
;Get the masks
;Set the MSE and RST 6.5 mask
;Write the masks
;Fake out the inline compares
;Leave

;Get the parity flag
;Store it in the right place
;Fake out the inline compares
;Leave

```

Error	Addr	Code	Seq	Source statement
			3179	
			3180	
			3181	
543D	3A	71B6	3182	FLIP: LDA EDPACK ;Check the cor
5440	FE	01	3183	CPI DEPCOD ;Is this a de sit?
5442	C4	5D1E	3184	CNZ SYNERR ;Error if not
5445	D3	A7	3185	OUT DISMEM ;Turn off memory
5447	2A	71BC	3186	LHLD EDDATA ;Get the address
544A	CD	521F	3187	CALL MCSEX1 ;Go use this subroutine
544D	2A	71BC	3188	LHLD EDDATA ;Get the UPC examine address
5450	22	41AA	3189	SHLD SWAP ;Put the corrected one back
5453	3A	41A9	3190	LDA NOPBUF+2 ;Get the bit to be flipped
5456	17		3191	RAL ;Put it in the carry
5457	3F		3192	CMC ;Flip it
5458	1F		3193	RAR ;Put it back
5459	32	416C	3194	STA TEMP1+2 ;Restore the byte
545C	2A	41A7	3195	LHLD NOPBUF ;Get the low two bytes
545F	22	416A	3196	SHLD TEMP1 ;Store them
5462	CD	4993	3197	CALL WRTNOP ;Go write a NOP again
5465	CD	49D4	3198	CALL LDUPC1 ;Go write the address again
5468	21	416A	3199	LXI H,TEMP1 ;Point to the storage area
546B	CD	5D05	3200	CALL WRTWCS ;Go write the data
546E	CD	49D4	3201	CALL LDUPC1 ;Restore the original UPC
5471	CD	49A7	3202	CALL LDCSR1 ;Restore the original CSR
5474	D3	A6	3203	OUT ENBMEM ;Turn memory back on
5476	3A	4196	3204	LDA CLKFLG ;Get the Clock Flag
5479	1F		3205	RAR ;Is it set?
547A	D2	547F	3206	JNC FLIP1 ;Skip setting the clock
547D	D3	21	3207	OUT SETCLK ;Turn the clock on
547F	AF		3208	FLIP1: XRA A ;Fake out the inline compares
5480	C9		3209	RET ;Leave
			3210	
			3211	
5481	3A	71B6	3212	UPCCHK: LDA EDPACK ;Get the command
5484	FE	00	3213	CPI EXMCOD ;Is this an Examine?
5486	CA	566E	3214	JZ UPCEXM ;If it is, jump
5489	D3	20	3215	OUT CLRCLK ;Stop the Base Machine Clock
548B	AF		3216	XRA A ;Make a 0
548C	32	4196	3217	STA CLKFLG ;Turn off the Clock flag
548F	D3	A7	3218	OUT DISMEM ;Turn off memory
5491	CD	4993	3219	CALL WRTNOP ;Write a NOP into the CSR
5494	21	71BC	3220	LXI H,EDDATA ;Point to the Data for the new UPC
5497	CD	49D1	3221	CALL LDUPC ;Write the new UPC
549A	D3	A6	3222	OUT ENBMEM ;Turn memory back on
549C	AF		3223	XRA A ;Fake out the inline compares
549D	C9		3224	RET ;Leave
			3225	
549E	CD	555B	3226	EDLS: CALL BEFORE ;Go prep things
54A1	3A	71B6	3227	LDA EDPACK ;Get the the packet header
54A4	FE	00	3228	CPI EXMCOD ;Is this an Examine?

Error	Addr	Code	Seq	Source statement	
S4A6	CA 54D1	3229	JZ	EXMLS	:Go examine the LS specified
S4A9	11 71BC	3230	LXI	D,EDDATA	:Point to the data
S4AC	CD 55CA	3231	CALL	WRTWRZ	:Go write it to WR 0
S4AF	21 7472	3232	LXI	H,SHFBUF	:Point to the shift buffer
S4B2	11 747D	3233	LXI	D,WRNLSN	:Point to the LSn to WRn uword
S4B5	06 03	3234	MVI	B,03	:Set up count for copying
S4B7	CD 00FB	3235	CALL	COPY1	:Go copy it
S4BA	2B	3236	DCX	H	:Decrement the pointer twice so it will
S4BB	2B	3237	DCX	H	:point to the second byte of the shift buffer
S4BC	3A 71B8	3238	LDA	EDADDR	:Get the LS address
S4BF	77	3239	MOV	M,A	:Store it in the second byte
S4C0	2B	3240	DCX	H	:Point to the low byte once again
S4C1	E5	3241	PUSH	H	:Save the address
S4C2	0E 03	3242	MVI	C,03	:Set up the count
S4C4	37	3243	STC		:Set the carry
S4C5	CD 4A00	3244	CALL	LSHIFT	:Go shift left one bit
S4C8	D1	3245	POP	D	:Put the address in DE
S4C9	CD 5649	3246	DEPLS1: CALL	EXECUT	:Go do the instruction
S4CC	CD 557C	3247	DEPLS2: CALL	AFTER	:Clean up to leave
S4CF	AF	3248	XRA	A	:Fake out the inline compares back in Deposit
S4D0	C9	3249	RET		:Leave
		3250			
S4D1	21 7472	3251	EXMLS: LXI	H,SHFBUF	:Point to the Shift buffer
S4D4	11 747A	3252	LXI	D,LSNWRN	:Point to the LSn to WRn uword
S4D7	06 03	3253	MVI	B,03	:Count for copying
S4D9	CD 00FB	3254	CALL	COPY1	:Go copy
S4DC	3A 71B8	3255	LDA	EDADDR	:Get the address
S4DF	21 7473	3256	LXI	H,SHFBUF+1	:Point to the second byte
S4E2	77	3257	MOV	M,A	:Put the address here
S4E3	2B	3258	DCX	H	:Point to the low byte once again
S4E4	E5	3259	PUSH	H	:Save the address
S4E5	0E 03	3260	MVI	C,03	:Number of bytes to shift accross
S4E7	37	3261	STC		:Set the carry
S4E8	CD 4A00	3262	CALL	LSHIFT	:Go shift the 3 bytes
S4EB	D1	3263	POP	D	:Put the address in DE
S4EC	CD 5649	3264	EXMLS1: CALL	EXECUT	:Go do this instruction
S4EF	21 71BF	3265	LXI	H,EDDATA+3	:Point to the Buffer for the data
S4F2	CD 55A4	3266	CALL	RDWRZ	:Go read WR 0
S4F5	CD 557C	3267	CALL	AFTER	:Clean up before leaving
S4F8	AF	3268	XRA	A	:Fake out the inline compares back in Examine
S4F9	C9	3269	RET		:Leave
		3270			
S4FA	CD 555B	3271	EDQ: CALL	BEFORE	:Go set some things up
S4FD	3A 71B6	3272	LDA	EDPACK	:Get the header byte of the packet
S500	FE 00	3273	CPI	EXMCD	:Is this an examine?
S502	CA 5511	3274	JZ	EXAMQ	:Jump if it is
S505	11 71BC	3275	LXI	D,EDDATA	:Point to the deposit data
S508	CD 55CA	3276	CALL	WRTWRZ	:Go write it to WR 0
S50B	11 7489	3277	LXI	D,WRTOQ	:Send it from WR 0 to Q
S50E	C3 54C9	3278	JMP	DEPLS1	:Go clean things up

Error	Addr	Code	Seq	Source statement	
			3279		
5511	11 7486		3280	EXAMQ: LXI D,QTOWR	;Point to the Q to WR 0 uword
5514	C3 54EC		3281	JMP EXMLS1	;Go clean things up
			3282		
5517	AF		3283	EDWR: XRA A	;Make a zero
5518	32 7478		3284	STA SOURCE	;Zero the source location
551B	32 7479		3285	STA DEST	;Zero the destination location
551E	CD 555B		3286	CALL BEFORE	;Go save some things
5521	3A 71B6		3287	LDA EDPACK	;Get the header byte of the packet
5524	FE 00		3288	CPI EXMCD	;Is this an examine?
5526	CA 554F		3289	JZ EXMWR	;Jump if it is
5529	3A 71B8		3290	LDA EDADDR	;Check for WR 0 address
552C	FE 10		3291	CPI 010H	;This is WR 0 address
552E	CA 5543		3292	JZ DEPWRO	;Go to some special code
5531	11 71BC		3293	LXI D,EDDATA	;Point to the data to be deposited
5534	CD 55CA		3294	CALL WRTWRZ	;Go write WR 0
5537	3A 71B8		3295	LDA EDADDR	;Get the address
553A	32 7479		3296	STA DEST	;Put it here
553D	CD 5604		3297	CALL MAKEWR	;Go make the WR 0 to WRn uword
5540	C3 54C9		3298	JMP DEPLS1	;Go clean things up
			3299		
5543	21 746E		3300	DEPWRO: LXI H,WRZBUF	;Point to the WR 0 buffer
5546	11 71BC		3301	LXI D,EDDATA	;Point to the deposit data
5549	CD 00F9		3302	CALL COPY	;Copy it over
554C	C3 54CC		3303	JMP DEPLS2	;This will write WR 0 and clean things up
			3304		
554F	3A 71B8		3305	EXMWR: LDA EDADDR	;Get the address
5552	32 7478		3306	STA SOURCE	;Put it here
5555	CD 5604		3307	CALL MAKEWR	;Go make the WRn to WR 0 uword
5558	C3 54EC		3308	JMP EXMLS1	;Go clean things up
			3309		
555B	D3 A7		3310	BEFORE: OUT DISMEM	;Turn off memory
555D	D3 20		3311	OUT CLRCLK	;Stop the clock
555F	D3 A1		3312	OUT DISPAR	;Turn off Halt on parity error
5561	21 4197		3313	LXI H,PARFLG	;Point to the Parity Flag
5564	7E		3314	MOV A,M	;Get it
5565	36 00		3315	MVI M,0	;Zero the PARFLG byte
5567	23		3316	INX H	;Point to a temp for it
5568	77		3317	MOV M,A	;Put it here
5569	21 41AA		3318	LXI H,SWAP	;Point to the swapping buffer
556C	CD 49D1		3319	CALL LDUPC	;Go get the UPC
556F	21 41A4		3320	LXI H,CSRBUF	;Point to the CSR buffer
5572	CD 49A4		3321	CALL LDCSR	;Go get the current CSR
5575	21 7471		3322	LXI H,WRZBUF+3	;Storage for WR 0
5578	CD 55A4		3323	CALL RDWRZ	;Go save WR 0
557B	C9		3324	RET	;Leave
			3325		
557C	11 746E		3326	AFTER: LXI D,WRZBUF	;Point to the WR 0 buffer
557F	CD 55CA		3327	CALL WRTWRZ	;Go write it
5582	21 41A4		3328	LXI H,CSRBUF	;Point to where the old CSR is

Error	Addr	Code	Seq	Source statement	
5585	CD 49A4	3329	CALL	LDCSR	;Go put it back
5588	21 41AA	3330	LXI	H,SWAP	;Get the old UPC
558B	CD 49D1	3331	CALL	LDUPC	;Put it back
558E	3A 4196	3332	LDA	CLKFLG	;See if the clock was going
5591	1F	3333	RAR		;If this sets the carry, it was
5592	D2 5597	3334	JNC	AFTER1	;Jump if not
5595	D3 21	3335	OUT	SETCLK	;Otherwise, turn on the clock
5597	D3 A6	3336	AFTER1: OUT	ENBMEM	;Turn on memory
5599	3A 4198	3337	LDA	PARTMP	;Check for Parity being on
559C	32 4197	3338	STA	PARFLG	;Put the Parity Flag back where it belongs
559F	1F	3339	RAR		;Will set the carry if it was
55A0	D0	3340	RNC		;Leave if not
55A1	D3 A0	3341	OUT	ENBPAR	;Turn on Halt on Parity
55A3	C9	3342	RET		;Now leave
		3343			
		3344			
55A4	22 746C	3345	RDWRZ: SHLD	PNTR2	;Save the address here
55A7	3E 04	3346	MVI	A,04	;Set up a count
55A9	32 747-	3347	STA	LPCNT2	;Save it
55AC	CD 5613	3348	RDWRZ1: CALL	WRR0T8	;Go rotate the WR 8 times
55AF	11 7483	3349	LXI	D,WRCRR	;Point to the U word that moves WR to CRR
55B2	CD 5649	3350	CALL	EXECUT	;Go do it
55B5	2A 746C	3351	LHLD	PNTR2	;Get the address
55B8	DB 80	3352	IN	READ	;Get the byte from the READ register
55BA	77	3353	MOV	M,A	;Store the byte
55BB	2B	3354	DCX	H	;Decrement the pointer
55BC	22 746C	3355	SHLD	PNTR2	;Save the updated address
55BF	3A 7477	3356	LDA	LPCNT2	;Get the loop count
55C2	3D	3357	DCR	A	;Decrement it
55C3	C8	3358	RZ		;Leave if zero
55C4	32 7477	3359	STA	LPCNT2	;Save the new value
55C7	C3 55AC	3360	JMP	RDWRZ1	;Loop back
		3361			
55CA	21 7472	3362	WRTWRZ: LXI	H,SHFBUF	;Point to the shift buffer
55CD	CD 00F9	3363	CALL	COPY	;Copy the data over
55D0	3E 20	3364	MVI	A,32	;Set up a count
55D2	32 7476	3365	STA	LPCNT1	;Put it here
55D5	11 748C	3366	LXI	D,CLRWR0	;Point to a uword to clear WR 0
55D8	CD 5649	3367	CALL	EXECUT	;Go do it
55DB	11 748F	3368	LXI	D,COMWR0	;Point to a uword to complement WR 0
55DE	CD 5649	3369	CALL	EXECUT	;Go do it
55E1	11 7492	3370	WRTWR1: LXI	D,ASLWR0	;Point to this Uword just in case
55E4	3A 7475	3371	LDA	SHFBUF+3	;Get the high byte
55E7	17	3372	RAL		;Check the carry
55E8	D2 55EE	3373	JNC	WRTWR2	;Jump if not set
55EB	11 7495	3374	LXI	D,RTLWR0	;Point to this uword
55EE	CD 5649	3375	WRTWR2: CALL	EXECUT	;Go do it
55F1	0E 04	3376	MVI	C,04	;Set up the number of bytes to shift accross
55F3	21 7472	3377	LXI	H,SHFBUF	;Point to where the data is
55F6	CD 4A00	3378	CALL	LSHIFT	;Go shift left one

Error	Addr	Code	Seq	Source statement	
	55F9	3A 7476	3379	LDA LPCNT1	;Get the loop count
	55FC	3D	3380	DCR A	;Decrement it
	55FD	C8	3381	RZ	;Leave if zero
	55FE	32 7476	3382	STA LPCNT1	;Save the new value if not
	5601	C3 55E1	3383	JMP WRTWR1	;Loop back
			3384		
	5604	21 7472	3385	MAKEWR: LXI H,SHFBUF	;Point to the Shift Buffer
	5607	E5	3386	PUSH H	;Save the address
	5608	11 7480	3387	LXI D,WRNWRN	;Point to the U word to be worked on
	560B	06 03	3388	MVI B,03	;Count for copying
	560D	CD 00FB	3389	CALL COPY1	;Go copy it over
	5610	3A 7478	3390	LDA SOURCE	;Get the source
	5613	E6 03	3391	ANI 03	;Mask off the unused bits
	5615	17	3392	RAL	;Rotate it left
	5616	17	3393	RAL	;Twice
	5617	47	3394	MOV B,A	;Save it for a bit
	5618	3A 7479	3395	LDA DEST	;Get the dest
	561B	E6 03	3396	ANI 03	;Mask off unused bits
	561D	B0	3397	ORA B	;OR in the source
	561E	2B	3398	DCX H	;Decrement the pointer
	561F	2B	3399	DCX H	;Once more to point to the middle byte
	5620	77	3400	MOV M,A	;Load the second byte
	5621	23	3401	INX H	;Fix the address
	5622	CD 5627	3402	CALL RSHF	;Go shift right one
	5625	D1	3403	POP D	;Put the address in DE
	5626	C9	3404	RET	;Leave
			3405		
	5627	0E 03	3406	RSHF: MVI C,03	;Prime the count
	5629	AF	3407	XRA A	;Clear the carries
	562A	7E	3408	RSHF1: MOV A,M	;GET BYTE
	562B	1F	3409	RAR	;PUT IT BACK
	562C	77	3410	MOV M,A	;Decrement the pointer
	562D	2B	3411	DCX H	;Decrement the count
	562E	0D	3412	DCR C	;Loop if not done
	562F	C2 562A	3413	JNZ RSHF1	;Leave when done
	5632	C9	3414	RET	
			3415		
			3416		
			3417		
	5633	3E 08	3418	WRTOT8: MVI A,08	;Set up the count
	5635	32 7476	3419	STA LPCNT1	;Store it
	5638	11 7495	3420	WRT8X: LXI D,RTLWR0	;Point to the Rotate Left U word
	563B	CD 5649	3421	CALL EXECUT	;Go do it
	563E	3A 7476	3422	LDA LPCNT1	;Get the count
	5641	3D	3423	DCR A	;Decrement it
	5642	C8	3424	RZ	;Leave if done
	5643	32 7476	3425	STA LPCNT1	;Save the count
	5646	C3 5638	3426	JMP WRT8X	;Loop until done
			3427		
			3428		

Error	Addr	Code	Seq	Source statement	
5649	21 41A7		3429	EXECUT: LXI	H,NOPBUF ;Point to the NOP buffer
564C	22 41A2		3430	SHLD	PNTR1 ;Save the address
564F	06 03		3431	MVI	B,03 ;Set up the count
5651	CD 00FB		3432	CALL	COPY1 ;Go copy the data
5654	21 41A7		3433	LXI	H,NOPBUF ;Point to the U word
5657	7E		3434	MOV	A,M ;Get the low byte
5658	23		3435	INX	H ;Point to the middle byte
5659	AE		3436	XRA	M ;XOR it in
565A	23		3437	INX	H ;Point to the high byte
565B	AE		3438	XRA	M ;XOR it in
565C	E2 5663		3439	JPO	EXE1 ;Jump on odd parity
565F	7E		3440	MOV	A,M ;Get the high byte if parity is even
5660	F6 80		3441	ORI	080H ;Set the MSB
5662	77		3442	MOV	M,A ;Put it back
5663	CD 49A7		3443	CALL	LDCSR1 ;Go load up the CSR
5666	D3 23		3444	OUT	SETSS ;Clock the machine
5668	D3 22		3445	OUT	CLRSS ;Reset the clock
566A	C9		3446	RET	;Leave
			3447		
			3448		
			3449		
566B	D3 20		3450	UPCEXM: OUT	CLRCLK ;Stop the Base machine clock
566D	21 4168		3451	LXI	H,TMPBUF ;Point to a temporary storage area
5670	CD 49D1		3452	CALL	LDUPC ;Go get the UPC
5673	11 4168		3453	LXI	D,TMPBUF ;Point to the Temp Buf again
5676	21 71BC		3454	LXI	H,EDDATA ;Destination address for the Data
5679	CD 00F9		3455	CALL	COPY ;Copy the Data over
567C	CD 49D4		3456	CALL	LDUPC1 ;Restore the UPC
567F	3A 4196		3457	LDA	CLKFLG ;Get the Clock Flag
5682	1F		3458	RAR	;Is the Clock Running Flag set
5683	D2 5688		3459	JNC	UPCEX1 ;Leave if it isn't
5686	D3 21		3460	OUT	SETCLK ;Restart the Clock if set
5688	AF		3461	UPCEX1: XRA	A ;Fake out any subsequent inline compares
5689	C9		3462	RET	;Leave
			3463		
			3464		
			3465		
			3466		
			3467		
			3468		
568A	C5		3469	COMFIN: PUSH	B ;Save the AND address for a while
568B	3A 7213		3470	LDA	FLAGS ;Going to check for an address
568E	E6 02		3471	ANI	AFLAG ;Is there an address?
5690	CC 5323		3472	CZ	ONEUP ;If no address, assume increment by one
5693	3A 7214		3473	LDA	FLAGS1 ;Check the + and - Flags.
5696	1F		3474	RAR	;Check the + Flag
5697	DC 5323		3475	CC	ONEUP ;If set, bump the address by one
569A	1F		3476	RAR	;Check the - Flag.
569B	DC 534F		3477	CC	ONEDN ;If set, decrement the address by one
569E	C1		3478	POP	B ;Get the AND address pointer back

Error	Addr	Code	Seq	Source statement
	569F	CD 56A5	3479	CALL ANDADR ;Save only the valid address bits
	56A2	C3 52FB	3480	JMP MERGE ;Go to some code that will do the work
			3481	;
			3482	;.....
			3483	
			3484	
			3485	
			3486	
	56A5	21 71B8	3487	ANDADR: LXI H,EDADDR ;Point to the address
	56A8	1E 04	3488	MVI E,04H ;Loop count
	56AA	0A	3489	ANDAD1: LDAX B ;Get the Mask
	56AB	A6	3490	ANA M ;AND it
	56AC	77	3491	MOV M,A ;Store it
	56AD	03	3492	INX B ;Bump the Mask pointer
	56AE	23	3493	INX H ;Bump the Address pointer
	56AF	1D	3494	DCR E ;Decrement the loop count
	56B0	C8	3495	RZ ;Leave if done
	56B1	C3 56AA	3496	JMP ANDAD1 ;Else, do it again
			3497	
			3498	
			3499	
	56B4	3A 7217	3500	RTEST: LDA RFLAG ;Get the R Flag
	56B7	1F	3501	RAR ;Is it set?
	56B8	D2 569C	3502	JNC RTEST1 ;Jump if not set
	56BB	E7	3503	RST 4 ;Go look for Control characters
			3504	;The return at the end of GETCHR will
			3505	;send us back to the command
	56BC	E1	3506	RTEST1: POP H ;Fix the stack (don't want to go back to the
			3507	;command)
	56BD	C9	3508	RET ;Leave
			3509	
			3510	
			3511	
			3512	;.....
			3513	;
	56BE	01 7215	3514	NTEST: LXI B,FLAGS2 ;Point to this flags byte
	56C1	0A	3515	LDAX B ;Get it
	56C2	A7	3516	ANA A ;See if the N Flag is set
	56C3	CA 56BC	3517	JZ RTEST1 ;If not, we're done
	56C6	0B	3518	DCX B ;Point to Flags 1
	56C7	0A	3519	LDAX B ;Get a copy
	56C8	E6 03	3520	ANI 03H ;Check for the - or + Flags
	56CA	C2 56D2	3521	JNZ NTEST1 ;If either set, skip the next part
	56CD	0B	3522	DCX B ;Point back at Flags
	56CE	0A	3523	LDAX B ;Get them
	56CF	E6 FD	3524	ANI 0FDH ;Get rid of the A flag
	56D1	02	3525	STAX B ;Store it
	56D2	21 720B	3526	NTEST1: LXI H,NBUFFR+3 ;Check for 0 count
	56D5	7E	3527	MOV A,M ;Get one byte of the count
	56D6	2B	3528	DCX H ;Decrement the pointer

Error	Addr	Code	Seq	Source statement
	56D7	B6	3529	ORA M ;OR in the next byte
	56D8	2B	3530	DCX H ;Decrement the pointer
	56D9	B6	3531	ORA M ;Get the next byte
	56DA	2B	3532	DCX H ;Decrement
	56DB	B6	3533	ORA M ;Next byte
	56DC	A7	3534	ANA A ;Is it Zero?
	56DD	01 7196	3535	LXI B,NEG1 ;Set up to subtract one from the count
	56E0	C2 6251	3536	JNZ ADDAD1 ;Jump if not zero
	56E3	E1	3537	POP H ;Fix up the stack
	56E4	AF	3538	XRA A ;Set the Z bit to indicate no errors
	56E5	C9	3539	RET ;Leave
			3540	;
			3541	;.....
			3542	
			3543	
			3544	
			3545	
			3546	
			3547	
			3548	;.....
			3549	;
			3550	;
			3551	Init Command and Init Subroutine
	56E6	CD 56F8	3552	INIT: CALL INISUB ;Call this subroutine to do the work
	56E9	21 7498	3553	LXI H,SPPACK ;Point to this packet
	56EC	CD 514E	3554	CALL SNDMSG ;Send it
	56EF	CD 5186	3555	CALL RCVMSX ;Get the reply
	56F2	CD 56B4	3556	CALL RTEST ;See if Repeat active
	56F5	C3 56E6	3557	JMP INIT ;If it is, do it again.
			3558	
	56F8	D3 AF	3559	INISUB: OUT CLRHLT ;Clear the Halt bit to the CPU
	56FA	D3 AD	3560	OUT CLRPMFI ;Reset the power fail interrupt
	56FC	D3 2C	3561	OUT CLRTMR ;Stop the Interval Timer
	56FE	20	3562	RIM ;Get the interrupt masks
	56FF	F6 09	3563	ORI 9H ;Set MSE and MS.5
	5701	30	3564	SIM ;Write the masks out
	5702	D3 A5	3565	OUT CLRTIM ;Clear the timer interrupt
	5704	CD 0055	3566	CALL TPINIT ;Go init the drive
	5707	21 0000	3567	LXI H,0 ;Make a word of 0
	570A	22 7070	3568	SHLD ISR ;Zero the ISR and IPR
	570D	22 4168	3569	SHLD TMPBUF ;Clear out this buffer for use further down
	5710	AF	3570	XRA A ;Make a 0 in Reg. A
	5711	32 7222	3571	STA BPHALT ;Zero this flag
	5714	32 707C	3572	STA ICCS ;Zero the ICCS location
	5717	32 7072	3573	STA RCVCSR ;Zero the RAM based terminal CSR
	571A	32 707D	3574	STA OLDRUN ;Zero the OLD RUN bit
	571D	32 707E	3575	STA ITSGL ;and the Interval Timer SGL flag
	5720	32 7099	3576	STA IMPXFR ;and the Implied XFR flag
	5723	32 40D7	3577	STA SPEND ;Clear the ^S pending flag
	5726	32 40D8	3578	STA QPEND ;and the ^Q pending flag

Error	Addr	Code	Seq	Source statement
	5729	21 7076	3579	LXI H,XMTCSR ;Point to the normal Xmit CSR
	572C	22 7078	3580	SHLD XCSADR ;Save the address here
	572F	23	3581	INX H ;Point to the normal Xmit DB
	5730	22 707A	3582	SHLD XDBADR ;Save the address here
	5733	D3 35	3583	OUT INITH ;Set Unibus Init
	5735	CD 5760	3584	CALL STAL10 ;Go stall for 10 Ms
	5738	D3 34	3585	OUT INITL ;Clear Unibus Init
	573A	D3 20	3586	OUT CLRCLK ;Stop the clock
	573C	D3 A7	3587	OUT DISMEM ;Turn off memory
	573E	CD 4993	3588	CALL WRTNOP ;Go write a NOP to the CSR
	5741	D3 A6	3589	OUT ENBMEM ;Turn on memory
	5743	21 4168	3590	LXI H,TMPBUF ;Point to this word of zeros
	5746	CD 49D1	3591	CALL LDUPC ;Go load it into the UPC
	5749	3E 01	3592	MVI A,1 ;Make a one
	574B	32 7074	3593	STA INTCSR ;Turn on the Xmit Ready in the Intermediate CSR
	574E	32 7076	3594	STA XMTCSR ;Turn on the Xmit Ready in the RAM based CSR
	5751	32 4196	3595	STA CLKFLG ;Set the clock flag so MAKCON will turn on the
			3596	clock when it finishes
	5754	CD 5E9E	3597	CALL MAKCON ;Go build the constants
	5757	21 71FB	3598	LXI H,INITPK ;Point to the Init Packet
	575A	CD 514E	3599	CALL SNDMSG ;Go send it to the CPU
	575D	C3 5186	3600	JMP RCVMSX ;Go get the reply
			3601	;Use the return at the end of RCVMSX
			3602	;
			3603	;.....
			3604	;
			3605	;
			3606	;
			3607	;.....
			3608	;
			3609	;
			3610	Stall routine
	5760	CD 5763	3611	STAL10: CALL STALLS ;We'll do this twice
	5763	01 041A	3612	STALLS: LXI B,1050 ;Load up a 5 Ms constant
	5766	0B	3613	1\$ DCX B ;Reduce the count
	5767	78	3614	MOV A,B ;Put one byte in A
	5768	B1	3615	ORA C ;OR in the other
	5769	C2 5766	3616	JNZ 1\$;Loop if not done
	576C	C9	3617	RET ;Leave when done
			3618	;
			3619	;.....
			3620	;
			3621	;
			3622	;
			3623	;.....
			3624	;
			3625	;
			3626	Boot Command
	576D	2A 40CA	3627	BOOT: LHLD BUFADR ;Get the pointer into the buffer
	5770	7E	3628	MOV A,M ;Get the next character

Error Addr	Code	Seq	Source statement	
5771	FE 0D	3629	CPI	CR
5773	CA 578D	3630	JZ	1\$
5776	FE 20	3631	CPI	' '
5778	C4 5D1E	3632	CNZ	SYNERR
577B	CD 4D23	3633	CALL	SPSTRP
577E	FE 3B	3634	CPI	'.'
5780	CA 578D	3635	JZ	1\$
5783	FE 21	3636	CPI	'!'
5785	CA 578D	3637	JZ	1\$
5788	FE 0D	3638	CPI	CR
578A	C2 5793	3639	JNZ	2\$
578D	CD 57D5	3640	1\$: CALL	BPSUB
5790	C3 6024	3641	JMP	CLDSTR
5793	CD 5925	3642	2\$: CALL	LODSU1
5796	3A 7210	3643	LDA	FNCNT
5799	A7	3644	ANA	A
579A	C2 57AD	3645	JNZ	3\$
579D	3A 7212	3646	LDA	DEVFLG
57A0	A7	3647	ANA	A
57A1	CC 5D1E	3648	CZ	SYNERR
57A4	CD 57D5	3649	CALL	BPSUB
57A7	3A 4106	3650	LDA	UNITX
57AA	C3 6027	3651	JMP	CLDST0
57AD	FE 03	3652	3\$: CPI	3
57AF	C4 5D1E	3653	CNZ	SYNERR
57B2	3A 7211	3654	LDA	EXTCNT
57B5	A7	3655	ANA	A
57B6	C4 5D1E	3656	CNZ	SYNERR
57B9	3A 4106	3657	LDA	UNITX
57BC	F6 30	3658	ORI	30H
57BE	32 726A	3659	STA	FOOFIL+3
57C1	21 726C	3660	LXI	H,FOOFIL+5
57C4	11 4107	3661	LXI	D,FNBUF
57C7	06 03	3662	MVI	B,3
57C9	CD 00FB	3663	CA	COPY1
57CC	CD 57D5	3664	C	BPSUB
57CF	21 7267	3665	LXI	H,FOOFIL
57D2	C3 602F	3666	JMP	CLDST1
		3667		
57D5	AF	3668	BPSUB: XRA	A
57D6	32 41B7	3669	STA	WARM
57D9	32 408A	3670	STA	COLD
57DC	32 40D4	3671	STA	NOTYPE
57DF	3C	3672	INR	A
57E0	32 40B6	3673	STA	PUBFLG
57E3	C9	3674	RET	
		3675		
		3676	;.....	
		3677		
		3678		

ZZ
.M
RA
Er

Error Addr Code Seq Source statement

```

3679
3680 ;.....
3681 ;
3682 ;      Bootstrap Prep
3683 ;
57E4 AF      3684 BSPREP: XRA      A      ;Make a 0
57E5 32 408A 3685          STA      COLD    ;Clear this flag
57E8 32 41B7 3686          STA      WARM    ;and this one
57EB 3C      3687          INR      A      ;Make a 1
57EC 32 40B6 3688          STA      PUBFLG   ;Set this to prevent going down into the ROM
57EF C3 6020 3689          JMP      CLDSTX   ;Go start things up
3690 ;
3691 ;.....
3692 ;
3693 ;.....
3694 ;
3695 ;
3696 ;      S Commands come here.
3697 ;
57F2      3698 S_COMMAND:
57F2 CD 4D2B 3699          CALL     PARSE      ;Go Parse the command string
57F5 3A 71B3 3700          LDA      NEWAS    ;Check for /C
57F8 FE 07   3701          CPI      07      ;Is it?
57FA C2 5815 3702          JNZ      START1    ;Jump if not WCS start
57FD D3 20   3703          OUT      CLRCLK   ;Stop the CPU Clock
57FF D3 A7   3704          OUT      DISMEM   ;Turn off memory
5801 CD 4993 3705          CALL     WRTNOP    ;Put a NOP in the CSR
5804 D3 A6   3706          OUT      ENBMEM   ;Turn memory on
5806 21 71FE 3707          LXI      H,ABUFFR  ;Point to the Address Buffer
5809 CD 49D1 3708          CALL     LDUPC    ;Go load the address into the UPC
580C 3E 01   3709 MICCON: MVI      A,01      ;Get a one
580E 32 4196 3710          STA      CLKFLG   ;Set the Clock Flag
5811 D3 21   3711          OUT      SETCLK   ;Start the CPU Clock
5813 AF      3712          XRA      A      ;Set the Z bit to signal no errors
5814 C9      3713          RET              ;Leave
3714
5815 FE 06   3715 START1: CPI      06      ;Check for /U
5817 CA 583C 3716          JZ       USTART    ;Go start the Micro
581A CD 56F8 3717          CALL     INISUB   ;Call the Init subroutine
581D 21 743C 3718          LXI      H,SADRS   ;Point to the address area
5820 11 71FE 3719          LXI      D,ABUFFR  ;Point to the address storage area
5823 3A 7214 3720          LDA      FLAGS1   ;Get the Flags1 Byte
5826 17      3721          RAL              ;See if the flag is set
5827 D2 582D 3722          JNC      START2    ;Jump if it isn't
582A 11 71BC 3723          LXI      D,EDDATA  ;Point to the last data instead
582D CD 00F9 3724 START2: CALL     COPY      ;Go copy it over
5830 21 7436 3725          LXI      H,STRPAK   ;Point to the start packet
5833 CD 514E 3726          CALL     SNDMSG    ;Go send a Deposit PC to the CPU
5836 CD 5186 3727          CALL     RCVMSX    ;Go get the message back
5839 C3 584B 3728          JMP      CONTO    ;Let the Continue code start up the machine

```

Error	Addr	Code	Seq	Source statement	
			3729		
583C	2A 71FE		3730	USTART: LHLD ABUFFR	;Get the low bytes of the Address Buffer
583F	E9		3731	PCHL	;Go there
			3732		
			3733	;	
			3734	;	
			3735	CONTINUE	
			3736	;	
5840	2A 40CA		3737	CONT: LHLD BUFADR	;Get the pointer into the Line buffer
5843	CD 4D23		3738	CALL SPSTRP	;Go strip off the spaces
5846	FE 0D		3739	CPI CR	;Look for Carriage Return
5848	C4 5D1E		3740	CNZ SYNERR	;Syntax error if not
584B	3E 01		3741	CONTO: MVI A,1	;Make a one
584D	32 40D3		3742	STA RUNFLG	;Set the run flag here
5850	3A 4155		3743	LDA CLKFLG	;Test the CPU Clock Flag
5853	1F		3744	RAR	;Is it set?
5854	DA 586B		3745	JC CNTIN1	;Jump if set
5857	21 7222		3746	LXI H,BPHALT	;Point to this flag
585A	7E		3747	MOV A,M	;Get it
585B	A7		3748	ANA A	;Is it zero?
585C	36 00		3749	MVI M,0	;Zero it
585E	C2 58E2		3750	JNZ CNTINO	;Jump if it wasn't
5861	3A 721E		3751	LDA STPFLG	;Get the Step Flag
5864	A7		3752	ANA A	;Is it set?
5865	CA 58DB		3753	JZ NOCON	;Jump if not set
5868	CD 580C		3754	CALL MICCON	;Go start up the Micro machine
586B	3A 721E		3755	CNTIN1: LDA STPFLG	;Get the Single Step flag
586E	32 71FD		3756	STA CONMOD	;Use it to set or clear the Modifier byte
5871	21 71FC		3757	LXI H,CONPAK	;Point to the continue packet
5874	CD 514E		3758	CALL SNDMSG	;Go send the packet
5877	CD 625F		3759	CNTIN2: CALL ENTER	;Go to Program I/O Mode
587A	D3 2C		3760	CONHLT: OUT CLRTMR	;Turn off the Interval Timer
587C	20		3761	RIM	;Get the interrupt masks
587D	F6 09		3762	ORI 9H	;Set MSE and MS.5
587F	30		3763	SIM	;Write the masks out
5880	3E 01		3764	MVI A,1	;Make a one for setting a flag
5882	32 707F		3765	STA HMPEND	;Set Halt Message Pending
5885	21 71DA		3766	LXI H,HLTBUF	;Point to a buffer area for halt information
5888	CD 5189		3767	CALL RCVMSG	;Go get a halt message from the CPU
588B	3A 40B5		3768	LDA PFFLAG	;Get the Power Fail Flag
588E	1F		3769	RAR	;See if it is set
588F	DC 5E0C		3770	CC ACFAI2	;Call if set
5892	21 71DB		3771	LXI H,HLTCOD	;Point to the Halt code
5895	7E		3772	MOV A,M	;Get it
5896	FE 02		3773	CPI 2	;Is it a ^P Halt?
5898	C2 58A3		3774	JNZ CONHL2	;Jump if not
589B	3A 40BE		3775	LDA LRMASK	;Get the Local/Remote Mask
589E	E6 40		3776	ANI 40H	;Look for APT in control
58A0	C2 58C1		3777	JNZ CONHL3	;Jump if APT in control
58A3	11 7416		3778	CONHL2: LXI D,HLTMSG+2	;Point to the destination for the ASCII

Error Addr	Code	Seq	Source statement
58A6	06 01	3779	MVI B,1 ;Set up a count
58A8	CD 5DBD	3780	CALL HEXASC ;Go convert the number over
58AB	3E 0D	3781	MVI A,CR ;Get a carriage return
58AD	DF	3782	RST 3 ;Go send it
58AE	21 7414	3783	LXI H,HLTMSG ;Point to the message
58B1	F7	3784	RST 6 ;Go send it
58B2	21 71DF	3785	LXI H,HALTPC+3 ;Point to the high byte of address
58B5	11 741E	3786	LXI D,PCADR ;Point to the PC Address buffer
58B8	06 04	3787	MVI B,04 ;Set up the count
58BA	CD 5DBD	3788	CALL HEXASC ;Go convert
58BD	21 7418	3789	LXI H,PCMSG ;Point to the PC Message
58C0	F7	3790	RST 6 ;Go print the message
58C1	AF	3791	CONHL3: XRA A ;Make a 0
58C2	32 707F	3792	STA HMPEND ;Clear out the Halt Message Pending flag
58C5	3A 721E	3793	LDA STPFLG ;Get the Single Step flag
58C8	1F	3794	RAR ;Put it in the carry
58C9	D8	3795	RC ;Leave if set
58CA	DB 02	3796	CONHL1: IN HLTFLG ;Get the Halt/Restart flag
58CC	1F	3797	RAR ;Put it in the carry
58CD	D2 4C32	3798	JNC IDLE ;Go to the Idle Loop if in Halt position
58D0	3A 71DB	3799	LDA HLTCOD ;Get the halt code
58D3	FE 02	3800	CPI 2 ;Check for a Control P halt
58D5	CA 4C32	3801	JZ IDLE ;Go to idle loop if it is
58D8	C3 6074	3802	JMP WRMSTR ;Go try to warm start the maching
		3803	
58DB	AF	3804	NOCON: XPA A ;Make a 0
58DC	32 40D3	3805	STA RUNFLG ;Turn this off
58DF	C3 580C	3806	JMP MICCON ;Go start up the Micro Machine
		3807	
58E2	CD 580C	3808	CNTINO: CALL MICCON ;Go start up the Micro Machine
58E5	C3 5877	3809	JMP CNTIN2 ;Go enter Program Mode
		3810	;
		3811	;.....
		3812	
		3813	
		3814	
		3815	;.....
		3816	;
58E8	2A 40CA	3817	HALT: LHLD BUFADR ;Get the pointer into command buffer
58EB	CD 4D23	3818	CALL SPSTRP ;Go Strip off the Spaces
58EE	FE 0D	3819	CPI CR ;Look for a carriage return
58F0	C4 5D1E	3820	CNZ SYNERR ;Go print error message
58F3	21 7414	3821	LXI H,HLTMSG ;Point to the Halt message
58F6	F7	3822	RST 6 ;Go Print it
58F7	21 7418	3823	LXI H,PCMSG ;Point to the PC of the halt
58FA	F7	3824	RST 6 ;Go print it
58FB	C9	3825	RET ;Leave
		3826	;
		3827	;.....
		3828	

Error	Addr	Code	Seq	Source statement
			3829	
			3830	
			3831	;.....
			3832	;
58FC	CD	590A	3833	LOAD: CALL LODSUB ;Go to the Load subroutine
58FF	CD	4500	3834	LOADZ: CALL LDXPRP ;Go prepare to enter the next part of load
5902	3A	4116	3835	LDA SUCCES ;Check the success code
5905	A7		3836	ANA A ;Is it 0?
5906	C4	5D2D	3837	CNZ TAPERR ;Go print the error message
5909	C9		3838	RET ;Leave with the Z bit set to signal no errors
			3839	
590A	CD	5925	3840	LODSUB: CALL LODSU1 ;Go figure things out
590D	3A	7213	3841	LDA FLAGS ;Get the Flags byte
5910	E6	0A	3842	ANI 0AH ;Save the N and A Flags
5912	FE	02	3843	CPI 02H ;Only the A Flag should be set
5914	C4	5D1E	3844	CNZ SYNERR ;Leave if not equal
5917	3A	7210	3845	LDA FNCNT ;Get the filename count
591A	A7		3846	ANA A ;Check it
591B	CC	5D1E	3847	CZ SYNERR ;Error if no filename specified
591E	3A	71B3	3848	LDA NEWAS ;Get the address space
5921	32	4115	3849	STA QUAL ;Put it here for the load code to use
5924	C9		3850	RET ;Leave
			3851	
5925	21	410E	3852	LODSU1: LXI H,EXTBUF ;Get the address of the Extension Buffer
5928	22	720E	3853	SHLD EXTPTR ;Store it in the Extension pointer
592B	21	4107	3854	LXI H,FNBUF ;Get the address of the File Name Buffer
592E	22	720C	3855	SHLD FNPTR ;Store it in the Pointer
5931	16	06	3856	MVI D,06 ;Prime the count
5933	CD	6AFC	3857	CALL ZBUF1 ;Zero this area
5936	21	410E	3858	LXI H,EXTBUF ;Point to the extension buffer
5939	CD	6AFA	3859	CALL ZROBUF ;Go do it
593C	21	4111	3860	LXI H,SBUFR ;Point to the Load Start address buffer
593F	CD	6AFA	3861	CALL ZROBUF ;Go zero it
5942	32	7212	3862	STA DEVFLG ;Clear out the Device Flag
5945	67		3863	MOV H,A ;Zero H
5946	6F		3864	MOV L,A ;Zero L
5947	22	7210	3865	SHLD FNCNT ;Zero the Filename and Extention counts
594A	3C		3866	INR A ;Make a one
594B	32	409B	3867	STA LODFLG ;Set the load flag
594E	3A	4150	3868	LDA DEFDRV ;Get the default drive number
5951	32	4106	3869	STA UNITX ;Put it here
5954	C3	4D2B	3870	JMP PARSE ;Go parse the command line
			3871	;
			3872	;.....
			3873	
			3874	
			3875	
			3876	;.....
			3877	;
5957	78		3878	MEMLOD: MOV A,B ;Get the character back into A

Error	Addr	Code	Seq	Source statement	
5958	21 4082	3879	LXI	H,BYTSUM	;Point to the checksum byte
5958	86	3880	ADD	M	;Add it in
595C	77	3881	MOV	M,A	;Send it out
595D	21 4193	3882	LXI	H,XFRBUF+1	;Point to a buffer area
5960	3A 40A7	3883	LDA	LODCNT	;Get the count
5963	1F	3884	RAR		;Check for odd or even byte
5964	D2 5971	3885	JNC	MEMLD1	;Jump if it is even
5967	70	3886	MOV	M,B	;Put the byte in ram
5968	06 01	3887	MVI	B,01	;Count for SNDMS1
596A	2B	3888	DCX	H	;Back up the pointer
596B	CD 5157	3889	CALL	SNDMS1	;Go send the data
596E	C3 4900	3890	JMP	TURCV2	;Go back
5971	2B	3891	MEMLD1: DCX	H	;Back up the pointer
5972	70	3892	MOV	M,B	;Send the data out
5973	C3 4900	3893	JMP	TURCV2	;Leave
		3894			
		3895			;.....
		3896			
		3897			
		3898			
		3899			;.....
		3900			
5976	CD 4D2B	3901	MICSTP: CALL	PARSE	;Go parse the command string
5979	D3 20	3902	OUT	CLRCLK	;Stop the CPU clock
597B	AF	3903	XRA	A	;Generate 0
597C	32 4196	3904	STA	CLKFLG	;Zero the Clock Flag
597F	3C	3905	INR	A	;Make a one
5980	32 721F	3906	STA	MICFLG	;Set this flag
5983	21 7213	3907	LXI	H,FLAGS	;Point to the Flags byte
5986	7E	3908	MOV	A,M	;Get it
5987	E6 40	3909	ANI	AACTIV	;Is there a count?
5989	CA 599F	3910	JZ	MICST3	;Jump if no count
598C	CD 5A0D	3911	CALL	FIXUP	;This is a Subroutine so Single Step can use it
598F	D3 23	3912	MICST1: OUT	SETSS	;Set single micro step
5991	D3 22	3913	OUT	CLRSS	;Clear single micro step
5993	CD 59B6	3914	CALL	GETUPC	;Go get and print the UPC
5996	CD 56BE	3915	CALL	NTEST	;This will check the count for us
5999	C3 598F	3916	JMP	MICST1	;Loop if count doesn't equal 0
		3917			
599C	CD 0046	3918	MICST2: CALL	GS_VECTOR	;Go fix up the silo pointers
599F	D3 23	3919	MICST3: OUT	SETSS	;Set Single Micro Step
59A1	D3 22	3920	OUT	CLRSS	;Then clear it
59A3	CD 59B6	3921	CALL	GETUPC	;Subroutine to get and print the UPC
59A6	CD 0049	3922	MICST4: CALL	RSVEC	;Go read from the silo
59A9	D2 59A6	3923	JNC	MICST4	;Loop if no character
59AC	78	3924	MOV	A,B	;Get the character into A
59AD	E6 7F	3925	ANI	7FH	;Clear the parity bit
59AF	FE 20	3926	CPI	' '	;Is it a space?
59B1	CA 599C	3927	JZ	MICST2	;Loop here
59B4	AF	3928	XRA	A	;Set the Z bit to signal no errors

Error	Addr	Code	Seq	Source statement	
	59B5	C9	3929	RET	;Leave
			3930		
			3931		
			3932		
59B6	21 41AA		3933	GETUPC: LXI H,SWAP	;Point to a temporary buffer area
59B9	CD 49D1		3934	CALL LDUPC	;Go get the UPC
59BC	2A 41AA		3935	LHLD SWAP	;Get the UPC
59BF	2B		3936	DCX H	;Decrement the value
59C0	22 4168		3937	SHLD TMPBUF	;Put the UPC here
59C3	CD 49D4		3938	CALL LDUPC1	;Go restore the UPC
59C6	21 4169		3939	LXI H,TMPBUF+1	;Prepare to convert from hex to ASCII
59C9	11 744B		3940	LXI D,UPCBUF	;Dest. is the Data segment of the print buffer
59CC	06 02		3941	MVI B,02	;Number of bytes to be converted
59CE	CD 5DBD		3942	CALL HEXASC	;Go convert
59D1	21 7440		3943	LXI H,UPCOUT	;Point to start of printout buffer
59D4	F7		3944	RST 6	;Go send out the line
59D5	C9		3945	RET	;Leave
			3946	:	
			3947	;	
			3948		
			3949		
			3950		
			3951	;	
			3952	:	
59D6	CD 4D2B		3953	STEP: CALL PARSE	;Go parse the rest of the command
59D9	21 721E		3954	LXI H,STPFLG	;Point the Step Flag byte
59DC	36 01		3955	MVI M,1	;Set it
59DE	21 7213		3956	LXI H,FLAGS	;See if there is a count
59E1	7E		3957	MOV A,M	;Get the FLAGS byte
59E2	E6 40		3958	ANI AACTIV	;The count will be in the Address buffer
59E4	CA 59F8		3959	JZ STEPB	;Jump if no count
59E7	CD 5A0D		3960	CALL FIXUP	;Go move the count to the N buffer
59EA	D3 AE		3961	STEPA: OUT SEThLT	;Turn on the Halt signal to the CPU
59EC	CD 584B		3962	CALL CONTO	;Go continue the machine
59EF	CD 56BE		3963	CALL NTEST	;Check the count
59F2	C3 59EA		3964	JMP STEPA	;Loop back if we return
			3965		
59F5	CD 0046		3966	STEPB: CALL GS_VECTOR	;Go fix up the silo pointers
59F8	D3 AE		3967	OUT SEThLT	;Turn on the Halt signal to the CPU
59FA	CD 584B		3968	CALL CONTO	;Go continue the machine
59FD	CD 0049		3969	STEPB1: CALL RSVEC	;Go look for characters in the silo
5A00	D2 59FD		3970	JNC STEPB1	;Loop if no character
5A03	78		3971	MOV A,B	;Get the character into A
5A04	E6 7F		3972	ANI 7FH	;Mask off the MSB
5A06	FE 20		3973	CPI ' '	;Is it a Space?
5A08	CA 59F5		3974	JZ STEPB	;Go step the machine
5A0B	AF		3975	XRA A	;Set the Z bit to signal no errors
5A0C	C9		3976	RET	;Leave if not a space
			3977		
			3978		

Error Addr	Code	Seq	Source statement
5A0D	21 7215	3979	FIXUP: LXI H,FLAGS2 ;Point the Flags 2 Byte
5A10	36 01	3980	MVI M,1 ;Set the N Flag
5A12	11 71FE	3981	LXI D,ABUFR ;Point to the count
5A15	21 7208	3982	LXI H,NBUFR ;Point to the Destination
5A18	C3 00F9	3983	JMP COPY ;Go copy
		3984	;
		3985	;
		3986	;
		3987	;
		3988	;
		3989	;
		3990	;
5A1B	3E 01	3991	INDRCT: MVI A,01 ;Get a 1
5A1D	32 7213	3992	STA FLAGS ;Set the space flag
5A20	CD 590A	3993	CALL LODSUB ;Let the load code do some of the work
5A23	21 4115	3994	INDRC1: LXI H,QUAL ;Point to the Qual byte
5A26	7E	3995	MOV A,M ;Get it
5A27	A7	3996	ANA A ;Should be zero
5A28	C4 5D1E	3997	CNZ SYNERR ;Error if not
5A2B	36 06	3998	MVI M,06 ;Force U space
5A2D	21 7600	3999	LXI H,INDBUF ;Point to the indirect command file buffer
5A30	22 4111	4000	SHLD SBUFR ;Save the address
5A33	22 721C	4001	SHLD INDADR ;Also put it in the Address Buffer
5A36	21 409D	4002	LXI H,INDFLG ;Point to the Indirect Command Flag
5A39	36 01	4003	MVI M,01 ;Set it
5A3B	C3 58FF	4004	JMP LOADZ ;Go share some code
		4005	;
		4006	;
		4007	;
		4008	;
		4009	;
		4010	;
		4011	;
		4012	TEST
		4013	;
		4014	;
		4015	;
		4016	;
		4017	;
		4018	;
		4019	;
		4020	;
		4021	;
		4022	;
		4023	;
5A3E	21 408B	4024	TEST: LXI H,CODFLG ;Point to this flag.
5A41	7E	4025	MOV A,M ;Get it
5A42	E6 01	4026	ANI 1 ;Clear bits 07-01
5A44	77	4027	MOV M,A ;Store this cleaned up version
5A45	0E 42	4028	MVI C,'B' ;Put an ASCII B in register C

Error	Addr	Code	Seq	Source statement	
	5A47	2A 40CA	4029	LHLD	BUFADR ;Get the current pointer value
	5A4A	CD 4D23	4030	CALL	SPSTRP ;Go strip out the spaces
			4031		;When we come back from SPSTRP, the character in
			4032		;A will be the non-space character we found in
			4033		;the buffer, and HL will point to that character
			4034		;in the buffer.
	5A4D	23	4035	INX	H ;Point to the next character
	5A4E	FE 0D	4036	CPI	CR ;See if the character SPSTRP found is a CR
	5A50	06 03	4037	MVI	B,3 ;In case it is, set up: Auto Mode, No Errors,
			4038		;Starting Micros, CRD.
	5A52	CA 5A74	4039	JZ	2\$;Go finish up and then Jump to the T_Vector
	5A55	FE 2F	4040	CPI	'/' ;Look for a Switch
	5A57	C4 5D1E	4041	CNZ	SYNERR ;Syntax Error if not
	5A5A	7E	4042	MOV	A,M ;Get the character after the Slash
	5A5B	23	4043	INX	H ;and point to the character after that.
	5A5C	FE 4D	4044	CPI	'M' ;Look for /M
	5A5E	06 09	4045	MVI	B,9H ;In case it is, set up: Menu Mode, No Errors,
			4046		;Not Starting Micros, CRD.
	5A60	CA 5A6C	4047	JZ	1\$;Jump if it is
	5A63	FE 43	4048	CPI	'C' ;Look for /C
	5A65	06 00	4049	MVI	B,0 ;In case it is, set up for NO CRD
	5A67	C4 5D1E	4050	CNZ	SYNERR ;Syntax Error if not T, T/M or T/C.
	5A6A	0E 41	4051	MVI	C,'A' ;Put an ASCII A in register C
	5A6C	CD 4D23	4052	CALL	SPSTRP ;Go read until non-space character found.
	5A6F	FE 0D	4053	CPI	CR ;Is it a Carriage Return?
	5A71	C4 5D1E	4054	CNZ	SYNERR ;Error if not
	5A74	3E 01	4055	MVI	A,1 ;Make a 1
	5A76	32 40BB	4056	STA	OFLAG ;Set the Control 0 Flag
	5A79	78	4057	MOV	A,B ;Get the CRD Flags Code
	5A7A	32 40FD	4058	STA	CRD_FLAGS ;Set up these flags
	5A7D	79	4059	MOV	A,C ;Get the correct n for ENKAN.EXE
	5A7E	32 4158	4060	STA	MICMON+4 ;Fix up the file name.
	5A81	78	4061	MOV	A,B ;Get the CRD flags back again.
	5A82	FE 09	4062	CPI	9H ;Look for Menu Mode.
	5A84	CA 5A98	4063	JZ	3\$;Jump if it is.
	5A87	20	4064	RIM	;Get the Interrupt Masks.
	5A88	F6 0B	4065	ORI	0BH ;Disable everything but Power Fail interrupts
	5A8A	30	4066	SIM	;Write the Masks
	5A8B	AF	4067	XRA	A ;Make a 0
	5A8C	32 4196	4068	STA	CLKFLG ;Turn off the Clock Flag
	5A8F	32 4197	4069	STA	PARFLG ;and the Parity Flag
	5A92	32 40D5	4070	STA	WCS_LOAD_FLAGS ;and the WCS flags
	5A95	C3 414D	4071	JMP	T_VECTOR ;Go figure things out
	5A98	AF	4072	XRA	A ;Make a 0
	5A99	32 408A	4073	STA	COLD ;Zero out this flag
	5A9C	3A 40D5	4074	LDA	WCS_LOAD_FLAGS ;Get the flags that indicate the state of the
	5A9F	1F	4075	RAR	;WCS and test for POWER.CMD executed.
	5AA0	D2 5F99	4076	JNC	COLD_BEGIN ;Jump if not
	5AA3	1F	4077	RAR	;How about CODE0n.CMD?
	5AA4	D2 5FD2	4078	JNC	COLD_BEGIN0 ;Jump if not.

Error Addr	Code	Seq	Source statement
SAA7	C3 5FE8	4079	JMP COLD_BEGIN1 ;If all the Micro Code is loaded, boot up.
		4080	;
		4081	;.....
		4082	
		4083	
		4084	
		4085	;.....
		4086	;
SAAA	E7	4087	WAICMD: RST 4 ;Go look for breakouts
SAAB	DB 83	4088	IN CPATTN ;Wait for CPU attention to be asserted
SAAD	1F	4089	RAR ;Is it asserted?
SAAE	D2 SAAA	4090	JNC WAICMD ;Loop back if not
SAB1	CD 5186	4091	CALL RCVMSX ;Go get the message
SAB4	21 749E	4092	LXI H, MEMADR ;Point to the dest
SAB7	11 71C2	4093	LXI D, RCVADR ;Point to the source
SABA	06 08	4094	MVI B, 08 ;Count for copying
SABC	CD 00FB	4095	CALL COPY1 ;Go copy
SABF	21 71C1	4096	LXI H, RCVMOD ;Point to the configuration number
SAC2	11 724C	4097	LXI D, CODFIL+5 ;Point to where it goes
SAC5	06 01	4098	MVI B, 1 ;Set up a count of one
SAC7	CD 5DBD	4099	CALL HEXASC ;Go convert it
SACA	CD 5186	4100	CALL RCVMSX ;Go get the second packet
SACD	21 74B4	4101	LXI H, MEMCOD ;Point to this dest
SAD0	11 71C2	4102	LXI D, RCVADR ;Point to this source
SAD3	CD 00F9	4103	CALL COPY ;Go copy it over
SAD6	AF	4104	XRA A ;Set the Z bit to signal no errors
SAD7	C9	4105	RET ;Leave
		4106	;
		4107	;.....
		4108	
		4109	
		4110	;.....
		4111	;
SAD8	3E 01	4112	XFER: MVI A, 1 ;Set the Control Character Disable flag
SADA	32 40FE	4113	STA CTLDIS ;Do it
SADD	CD 0046	4114	XFER1: CALL GS_VECTOR ;Go get the checksum
SAE0	D2 SADD	4115	JNC XFER1 ;Loop until a character is received
SAE3	3A 4082	4116	LDA BYTSUM ;Get the checksum
SAE6	A7	4117	ANA A ;Check for 0
SAE7	C4 5D4E	4118	CNZ NACK_X_CMND ;Error if result is not zero
SAEA	CD 4D2B	4119	CALL PARSE ;Go parse the command
SAED	21 418A	4120	LXI H, XFRADR ;Point to a place where the address goes
SAF0	11 71FE	4121	LXI D, ABUFFR ;Point to where the address was stored
SAF3	06 08	4122	MVI B, 8 ;Count of 8
SAF5	CD 00FB	4123	CALL COPY1 ;Go move it over
SAF8	CD 623C	4124	CALL XFRCHK ;Check for 0 count
SAFB	CC 5D1E	4125	CZ SYNERR ;Can't have 0 byte transfers
SAFE	3A 71B3	4126	LDA NEWAS ;Get the Address space byte
SB01	FE 06	4127	CPI 6 ;See if 8085 Ram space was specified
SB03	CA 5BC2	4128	JZ XFRRAM ;Jump if it is

Error	Addr	Code	Seq	Source statement	
SB06	FE 07	4129	CPI 7		;How about WCS?
SB08	CA 5C27	4130	JZ XFRWCS		;Jump if it is
SB08	FE 02	4131	CPI 2		;See if P or V
SB0D	D4 5D1E	4132	CNC SYNERR		;Error if not
SB10	21 4188	4133	LXI H,XFRPAK		;Point to the Transfer Packet
SB1	CD 514E	4134	CALL SNDMSG		;Go send it
SB16	CD 5CE7	4135	CALL XFRCOM		;Go share some code
SB19	DA 5B7B	4136	JC XMEMO		;When we return, the carry will either be set
		4137			;or clear. If set it is an X Out
SB1C	CD 5CAC	4138	XMEMI: CALL GETDAT		;Go try for some data
SB1F	CA 5B51	4139	JZ XMEMI2		;If we return with the Z bit set, count=0
SB22	D3 CC	4140	OUT WRITE		;Send the data out
SB24	D3 AA	4141	OUT SETATN		;Raise the Console Attention signal
SB26	0E 00	4142	MVI C,0		;Set up a timeout
SB28	DB 82	4143	XMEMH: IN CPUACK		;Look for the ack
SB2A	1F	4144	RAR		;Rotate it into the carry
SB2B	DA 5B35	4145	JC XMEMH1		;Jump if set
SB2E	0D	4146	DCR C		;Reduce the count
SB2F	C2 5B28	4147	JNZ XMEMH		;Loop back if not yet zero
SB32	C3 5B48	4148	JMP XMEMI1		;Jump if timeout
SB35	CD 5CAC	4149	XMEMH1: CALL GETDAT		;Go look for another byte
SB38	D3 CC	4150	OUT WRITE		;Write it to the CPU
SB3A	D3 AB	4151	OUT CLRATN		;Clear Console Attention
SB3C	0E 00	4152	MVI C,0		;Set up a time out
SB3E	DB 82	4153	XMEML: IN CPUACK		;Look for CPU ack
SB40	1F	4154	RAR		;Put it in the carry
SB41	D2 5B1C	4155	JNC XMEMI		;Jump if it has gone away
SB44	0D	4156	DCR C		;Reduce the count
SB45	C2 5B3E	4157	JNZ XMEML		;Loop if not yet zero
SB48	CD 5CAC	4158	XMEMI1: CALL GETDAT		;Get all the data
SB4B	C2 5B48	4159	JNZ XMEMI1		;Loop until we get it all
SB4E	CD 5D5E	4160	CALL COMERR		;Go report the error
		4161			
SB51	CD 5186	4162	XMEMI2: CALL RCVMSX		;Go get the reply packet from the CPU
SB54	C2 5B61	4163	JNZ XMEMI3		;Jump if not good
SB57	3A 71C6	4164	LDA RCVDAT		;Get the checksum the microcode calculated
SB5A	21 4082	4165	LXI H,BYTSUM		;Point to mine
SB5D	BE	4166	CMP M		;Are they the same
SB5E	CA 5B67	4167	JZ GETCS		;Jump if good
SB61	CD 5B67	4168	XMEMI3: CALL GETCS		;Go get the checksum character
SB64	CD 5D5E	4169	CALL COMERR		;Go send an error message
		4170			
SB67	CD 0046	4171	GETCS: CALL GS_VECTOR		;Go get the checksum character
SB6A	D2 5B67	4172	JNC GETCS		;Loop until a character comes in
SB6D	3A 4082	4173	LDA BYTSUM		;Get the checksum
SB70	A7	4174	ANA A		;Check it for 0
SB71	C4 5D56	4175	CNZ NACK_X_DATA		;Error if not
SB74	32 40D4	4176	STA NOTYPE		;Clear the typeout inhibit flag
SB77	32 40FE	4177	STA CTLDIS		;Allow control characters to function now
SB7A	C9	4178	RET		;and go back

Error Addr	Code	Seq	Source statement
		4179	
5B7B	21 4192	4180	XMEM0: LXI H,XFRBUF ;Point to this buffer area
5B7E	06 01	4181	MVI B,1 ;Count for two byte transfer
5B80	CD 518E	4182	CALL RCVMS1 ;Go get the data from the CPU
5B83	CA 5B94	4183	JZ XMEM02 ;Jump if OK
5B86	CD 5CD9	4184	XMEM04: CALL SNDDAT ;Send garbage until count=0
5B89	C2 5B86	4185	JNZ XMEM04 ;Loop until 0
5B8C	3A 4082	4186	LDA BYTSUM ;Get the byte checksum
5B8F	2F	4187	CMA ;Complement
5B90	DF	4188	RST 3 ;Send it
5B91	CD 5D5E	4189	CALL COMERR ;Send an error message
5B94	CD 5CD9	4190	XMEM02: CALL SNDDAT ;Go send the data
5B97	CA 5BA6	4191	JZ XMEM01 ;Jump if count =0
5B9A	3A 4193	4192	LDA XFRBUF+1 ;Get the second byte
5B9D	32 4192	4193	STA XFRBUF ;Put it here
5BA0	CD 5CD9	4194	CALL SNDDAT ;Go check the count and send the data
5BA3	C2 5B7B	4195	JNZ XMEM0 ;Loop back if not 0 count
5BA6	CD 5186	4196	XMEM01: CALL RCVMSX ;Go get the packet from the CPU
5BA9	21 71C6	4197	LXI H,RCVDAT ;Point to the checksum he calculated
5BAC	3A 4082	4198	LDA BYTSUM ;Get mine
5BAF	BE	4199	CMP M ;Are they equal?
5BB0	CA 5BB7	4200	JZ XOFIN ;IF CORRECT, THEN go finish up cleanly.
5BB3	3D	4201	DCR A ;ELSE, screw up the checksum
5BB4	32 4082	4202	STA BYTSUM ;to cause an error.
5BB7	3A 4082	4203	XOFIN: LDA BYTSUM ;Get the checksum
5BBA	2F	4204	CMA ;Complement it
5BBB	3C	4205	INR A ;Make the twos complement
5BBC	DF	4206	RST 3 ;Go send it out
5BBD	AF	4207	XRA A ;Set the Z bit to indicate no errors
5BBE	32 40FF	4208	STA NOCHK ;Zero this out
5BC1	C9	4209	RET ;Leave
		4210	
		4211	
5BC2	11 71FE	4212	XFRRAM: LXI D,ABUFFR ;Point to where the address is
5BC5	21 71AA	4213	LXI H,RAMADR ;Point to this address
5BC8	CD 6242	4214	CALL CMPCHK ;Go see if it is the same
5BCB	C2 5BDA	4215	JNZ XFRA1 ;Jump if not
5BCE	21 71AE	4216	LXI H,RAMCNT ;Point to this count
5BD1	11 7202	4217	LXI D,DBUFFR ;Point to the count specified
5BD4	CD 6242	4218	CALL CMPCHK ;Go see if equal
5BD7	CA 000B	4219	JZ XVEC ;Go down into the ROM
5BDA	CD 5CE7	4220	XFRA1: CALL XFRCOM ;Go share some code
5BDD	DA 5C01	4221	JC XRAM0 ;Jump if direction is Out
5BE0	CD 5CAC	4222	XRAM1: CALL GETDAT ;Go get a byte
5BE3	CA 5B67	4223	JZ GETCS ;If count=0, get the checksum and check it
5BE6	2A 418A	4224	LHLD XFRADR ;Get the address
5BE9	7C	4225	MOV A,H ;Put the high byte in A
5BEA	FE F0	4226	CPI 0F0H ;Check for 8085 I/O address
5BEC	CA 5BF7	4227	JZ XRAM13 ;Jump if it is
5BEF	70	4228	MOV M,B ;Send the data out to memory

Error Addr	Code	Seq	Source statement	
5BF0	23	4229	XRAMI2: INX H	;Bump the address
5BF1	22 418A	4230	SHLD XFRADR	;Save the updated address
5BF4	C3 5BE0	4231	JMP XRAMI1	;Loop back
		4232		
5BF7	7D	4233	XRAMI3: MOV A,L	;Get the I/O address
5BF8	32 5BFD	4234	STA XRAMI4+1	;Alter I stream
5BFB	78	4235	MOV A,B	;Get the data into A
5BFC	D3 84	4236	XRAMI4: OUT UPC14	;Dummy address
5BFE	C3 5BF0	4237	JMP XRAMI2	;Rejoin normal XRAMI flow
		4238		
		4239		
		4240		
		4241		
5C01	CD 623C	4242	XRAM0: CALL XFRCHK	;Go check for 0 count
5C04	CA 5BB7	4243	JZ XOFIN	;Go finish things up if it is
5C07	2A 418A	4244	LHLD XFRADR	;Get the specified address
5C0A	7C	4245	MOV A,H	;Get the high byte
5C0B	FE F0	4246	CPI 0F0H	;See if I/O space is specified
5C0D	CA 5C1E	4247	JZ XRAM02	;Jump if it is
5C10	7E	4248	MOV A,M	;Get the data
5C11	32 4192	4249	XRAM01: STA XFRBUF	;Save the data here
5C14	23	4250	INX H	;Bump the address pointer
5C15	22 418A	4251	SHLD XFRADR	;Save the updated address
5C18	CD 5CD9	4252	CALL SNDDAT	;Go send it to APT
5C1B	C3 5C01	4253	JMP XRAM0	;Loop back
		4254		
5C1E	7D	4255	XRAM02: MOV A,L	;Get the I/O address
5C1F	32 5C23	4256	STA XRAM03+1	;Alter I stream
5C22	DB 84	4257	XRAM03: IN UPC14	;Dummy address
5C24	C3 5C11	4258	JMP XRAM01	;Jump back
		4259		
		4260		
5C27	D3 20	4261	XFRWCS: OUT CLRCLK	;Turn off the CPU clock
5C29	CD 4993	4262	CALL WRTNOP	;NOP the CSR
5C2C	21 418A	4263	LXI H,XFRADR	;Point to the address
5C2F	CD 49D1	4264	CALL LDUPC	;Go load it
5C32	CD 5CE7	4265	CALL XFRCOM	;Go share some code
5C35	DA 5C64	4266	JC XWCS0	;Jump if set (indicates X out)
5C38	CD 5CAC	4267	XWCS1: CALL GETDAT	;Go look for a byte
5C3B	CA 5C5B	4268	JZ XWCS11	;Will come back with Z set if count=0
5C3E	D3 08	4269	OUT WCSB0	;Write byte 0
5C40	CD 5CAC	4270	CALL GETDAT	;Get the next byte
5C43	CA 5C5B	4271	JZ XWCS11	;Jump if count = 0
5C46	D3 09	4272	OUT WCSB1	;Write byte 1
5C48	CD 5CAC	4273	CALL GETDAT	;Get the next byte
5C4B	CA 5C5B	4274	JZ XWCS11	;Jump if count = 0
5C4E	D3 0A	4275	OUT WCSB2	;Write byte 2
5C50	D3 37	4276	OUT SETWCK	;Set the WCS clock
5C52	D3 36	4277	OUT CLRWCK	;Clear it
5C54	D3 27	4278	OUT SETUPK	;Set the the UPC clock

Error	Addr	Code	Seq	Source statement	
	5C56	D3 26	4279	OUT CLRUPK	;Clear it
	5C58	C3 5C38	4280	JMP XWCSI	;Loop back
			4281		
	5C5B	CD 5B67	4282	XWCSI1: CALL GETCS	;Go get the checksum
	5C5E	CD 4993	4283	CALL WRTNOP	;NOP the CSR
	5C61	C3 5CA1	4284	JMP XWCS03	;Go share some code
			4285		
	5C64	21 4168	4286	XWCS0: LXI H,TMPBUF	;Point to this temp
	5C67	11 41A7	4287	LXI D,NOPBUF	;Point to the CSR we want to save
	5C6A	06 03	4288	MVI B,3	;Count
	5C6C	CD 00FB	4289	CALL COPY1	;Go move the data over
	5C6F	D3 23	4290	XWCS01: OUT SETSS	;Single step the machine
	5C71	D3 22	4291	OUT CLRSS	;Reset the single step clock
	5C73	CD 4993	4292	CALL WRTNOP	;Go get the data and rewrite the NOP
	5C76	3A 41A7	4293	LDA NOPBUF	;Get the low byte
	5C79	32 4192	4294	STA XFRBUF	;Put it here
	5C7C	CD 5CD9	4295	CALL SNDDAT	;Go send it
	5C7F	3A 41A8	4296	LDA NOPBUF+1	;Get the next byte
	5C82	32 4192	4297	STA XFRBUF	;Put it here
	5C85	CD 5CD9	4298	CALL SNDDAT	;Go send it
	5C88	3A 41A9	4299	LDA NOPBUF+2	;Get the byte
	5C8B	32 4192	4300	STA XFRBUF	;Save it here
	5C8E	CD 5CD9	4301	CALL SNDDAT	;Go send it
	5C91	CD 623C	4302	CALL XFRCHK	;Go check the Transfer count
	5C94	C2 5C6F	4303	JNZ XWCS01	;Loop back if not 0
	5C97	3A 4082	4304	LDA BYTSUM	;Get my checksum
	5C9A	DF	4305	RST 3	;Go send it
	5C9B	21 4168	4306	XWCS02: LXI H,TMPBUF	;Point to the old CSR
	5C9E	CD 49A4	4307	CALL LDCSR	;Go write it
	5CA1	CD 49D4	4308	XWCS03: CALL LDUPC1	;Go load the old UPC
	5CA4	3A 4196	4309	LDA CLKFLG	;Get the clock flag
	5CA7	A7	4310	ANA A	;Is it set?
	5CA8	C8	4311	RZ	;Leave if not
	5CA9	D3 21	4312	OUT SETCLK	;Turn the clock back on otherwise
	5CAB	C9	4313	RET	;Leave
			4314		
			4315		
	5CAC	21 4191	4316	GETDAT: LXI H,XFRCNT+3	;Point to the high byte of the count
	5CAF	7E	4317	MOV A,M	;Get it
	5CB0	2B	4318	DCX H	;Back the pointer up
	5CB1	B6	4319	ORA M	;OR in the next byte
	5CB2	2B	4320	DCX H	;Keep backing it up
	5CB3	B6	4321	ORA M	;and ORing things in
	5CB4	2B	4322	DCX H	;Last one
	5CB5	B6	4323	ORA M	;Whew!
	5CB6	C8	4324	RZ	;Leave if 0
	5CB7	AF	4325	XRA A	;Clear the carry
	5CB8	01 7196	4326	LXI B,NEG1	;Point to a minus 1
	5CBB	0A	4327	ONE: LDAX B	;Get the low byte of the constant
	5CBC	8E	4328	ADC H	;Add it to the address

Error	Addr	Code	Seq	Source statement	
	SCBD	77	4329	MOV M,A	;Store it
	SCBE	03	4330	INX B	;Bump the constant pointer
	SCBF	23	4331	INX H	;Bump the address pointer
	SCC0	0A	4332	TWO: LDAX B	;Get the low byte of the constant
	SCC1	8E	4333	ADC M	;Add it to the address
	SCC2	77	4334	MOV M,A	;Store it
	SCC3	03	4335	INX B	;Bump the constant pointer
	SCC4	23	4336	INX H	;Bump the address pointer
	SCC5	0A	4337	THREE: LDAX B	;Get the low byte of the constant
	SCC6	8E	4338	ADC M	;Add it to the address
	SCC7	77	4339	MOV M,A	;Store it
	SCC8	03	4340	INX B	;Bump the constant pointer
	SCC9	23	4341	INX H	;Bump the address pointer
	SCCA	0A	4342	FOUR: LDAX B	;Get the low byte of the constant
	SCCB	8E	4343	ADC M	;Add it to the address
	SCCC	77	4344	MOV M,A	;Store it
	SCCD	03	4345	INX B	;Bump the constant pointer
	SCCE	23	4346	INX H	;Bump the address pointer
	SCCF	CD 0046	4347	GETDA1: CALL GS_VECT'	;Go get the next character
	SCD2	D2 SCCF	4348	JNC GETDA1	;Wait for a character
	SCD5	AF	4349	XRA A	;Make a 0
	SCD6	3C	4350	INR A	;Clear the Z bit
	SCD7	78	4351	MOV A,B	;Get the character back
	SCD8	C9	4352	RET	;Leave with the Z bit clear
			4353		
	SCD9	21 418E	4354	SNDDAT: LXI H,XFRCNT	;Point to the count again
	SCDC	01 7196	4355	LXI B,NEG1	;Point to a longword -1
	SCDF	CD 6251	4356	CALL ADDAD1	;Go bump the count
	SCE2	3A 4192	4357	LDA XFRBUF	;Get the byte to be sent
	SCE5	DF	4358	RST 3	;Go send it
	SCE6	C9	4359	RET	;Leave
			4360		
	SCE7	21 4203	4361	XFRCOM: LXI H,PROMPT	;Point to the prompt
	SCEA	F7	4362	RST 6	;Send an ACK
	SCEB	AF	4363	XRA A	;Make a 0
	SCEC	32 4082	4364	STA BYTSUM	;Zero out the checksum
	SCEF	21 4191	4365	LXI H,XFRCNT+3	;Point to the high byte of the count
	SCF2	46	4366	MOV B,M	;Get a copy into B
	SCF3	78	4367	MOV A,B	;Put a copy in A
	SCF4	E6 7F	4368	ANI 7FH	;Clear the MSB
	SCF6	77	4369	MOV M,A	;Put it back into XFRCNT+3
	SCF7	78	4370	MOV A,B	;Get the unmodified copy
	SCF8	17	4371	RAL	;Put the MSB in the carry
	SCF9	D0	4372	RNC	;Leave if transfer is inward
	SCFA	AF	4373	XRA A	;Make a 0
	SCFB	32 40FE	4374	STA CTLDIS	;Clear this flag
	SCFE	3C	4375	INR A	;Make a 1
	SCFF	32 40FF	4376	STA NOCHK	;and set this one
	SD02	C9	4377	RET	;Leave
			4378		

Error	Addr	Code	Seq	Source statement
			4379	
			4380	
			4381	;
			4382	;.....
			4383	
SD03	AF		4384	CRCMD: XRA A ;Set the Z bit to signal no errors
SD04	C9		4385	RET ;Leave
			4386	
			4387	
			4388	;.....
			4389	;
			4390	;THE FOLLOWING ROUTINE WRITES ONE UWORD.
			4391	;HL POINTS TO THE FIRST BYTE TO BE WRITTEN
			4392	;
SD05	7E		4393	WRTWCS: MOV A,M ;A gets CSR <7:0>
SD06	D3 08		4394	OUT WCSB0 ;Write byte 0
SD08	23		4395	INX H ;INCREMENT POINTER
SD09	7E		4396	MOV A,M ;A GETS CSR<15:8>
SD0A	D3 09		4397	OUT WCSB1 ;Write byte 1
SD0C	23		4398	INX H ;INCREMENT POINTER
SD0D	7E		4399	MOV A,M ;A GETS CSR<23:16>
SD0E	D3 0A		4400	OUT WCSB2 ;Write byte 2
SD10	23		4401	INX H ;INCREMENT POINTER
SD11	D3 37		4402	OUT SETWCK ;Set the clock
SD13	D3 36		4403	OUT CLRWCK ;Reset it
SD15	C9		4404	RET ;RETURN
			4405	;
			4406	;.....
			4407	
			4408	
			4409	;.....
			4410	;
SD16			4411	CF_ERROR:
SD16	3E 02		4412	MVI A,CFERR_CODE ;Error code
SD18	11 738B		4413	LXI D,CF_TL_MSG ;Point to this message
SD1B	C3 5D80		4414	JMP ERROR ;Go service it
			4415	
SD1E	AF		4416	SYNEPR: XRA A ;Make a 0
SD1F	32 4182		4417	STA CMDVAL ;Zero the Command Valid Flag
SD22	32 7000		4418	STA CHRCNT ;Zero the character count
SD25	3E 03		4419	MVI A,SYN_CODE ;Error code
SD27	11 7352		4420	LXI D,SYNMSG ;Point to this error message
SD2A	C3 5D80		4421	JMP ERROR ;Go print the error message
			4422	
SD2D	AF		4423	TAPERR: XRA A ;Make a zero
SD2E	32 40BB		4424	STA OFLAG ;Zero the O flag
SD31	32 409C		4425	STA DIRFLG ;Zero this flag
SD34	32 409B		4426	STA LODFLG ;and this one
SD37	32 409D		4427	STA INDFLG ;and this one also
SD3A	32 40FE		4428	STA CTLDIS ;Allow control characters

Error Addr	Code	Seq	Source statement	
SD3D	CD 4266	4429	CALL MYLDER	;Go print the error information
SD40	3E 0D	4430	MVI A,CR	;Get a carriage return
SD42	DF	4431	RST 3	;Go send it
SD43	C3 5C A2	4432	JMP ERROR1	;Go share some code
		4433		
SD46	3E 04	4434	MEMERR: MVI A, MEM_CODE	;Error code
SD48	11 7365	4435	LXI D, MEMMSG	;Point to the memory error message
SD4B	C3 5D80	4436	JMP ERROR	;Go print it
		4437		
SD4E		4438	NACK_X_CMND:	
SD4E	3E 07	4439	MVI A, NACKCHND_CODE	;Code for Command Checksum Error
SD50	11 73A7	4440	LXI D, X_CMND_MSG	;Error message address
SD53	C3 5D80	4441	JMP ERROR	;Go report the error
		4442		
SD56		4443	NACK_X_DATA:	
SD56	3E 08	4444	MVI A, NACKDATA_CODE	;Code for Data Checksum Error
SD58	11 73C6	4445	LXI D, X_DATA_MSG	;Error message address
SD5B	C3 5D80	4446	JMP ERROR	;Go report the error
		4447		
SD5E	D3 AB	4448	COMERR: OUT CLRATN	;Clear attention
SD60	D3 A9	4449	OUT CLRACK	;Clear ack
SD62	D3 20	4450	OUT CLRCLK	;Turn off the CPU clock
SD64	AF	4451	XRA A	;Make a 0
SD65	32 4196	4452	STA CLKFLG	;Turn off this flag
SD68	3E 05	4453	MVI A, COMM_CODE	
SD6A	11 7378	4454	LXI D, COMMMSG	;Point to the Communications error message
SD6D	C3 5D80	4455	JMP ERROR	;Go print it
		4456		
SD70	3E 06	4457	CLKERR: MVI A, CLOCK_CODE	
SD72	11 73E2	4458	LXI D, CLKMSG	;Point to the Clock Stopped message
SD75	C3 5D80	4459	JMP ERROR	;Go report it
		4460		
SD78		4461	CRD_ERROR:	
SD78	3E 01	4462	MVI A, CRD_CODE	;Generate an Error Code
SD7A	11 73FA	4463	LXI D, CRD_ERR_MSG	;Point to this error message.
SD7D	C3 5D80	4464	JMP ERROR	;Go report it
		4465		
SD80	32 4116	4466	ERR: STA SUCCES	;Write the error code
SD83	E1	4467	POP H	;Get the error PC
SD84	22 40B9	4468	SHLD MYPC	;Save it
SD87	3A 40D4	4469	LDA NOTYPE	;Get the No Timeout Flag
SD8A	A7	4470	ANA A	;Is it set?
SD8B	C2 5DA2	4471	JNZ ERROR1	;Jump if set
SD8E	EB	4472	XCHG	;Get the message address into HL
SD8F	AF	4473	XRA A	;Make a 0
SD90	32 40FE	4474	STA CTLDIS	;Turn off the control character disable flag
SD93	32 40BB	4475	STA OFLAG	;Clear the O Flag
SD96	32 409D	4476	STA INDFLG	;Clear the Indirect Command Flag
SD99	F7	4477	RST 6	;Go print the message
SD9A	3A 4116	4478	LDA SUCCES	;Get the Success code

Error	Addr	Code	Seq	Source statement
SD9D	FE 05	4479	CPI	COMM_CODE ;Is is a comm error?
SD9F	CC 59B6	4480	CZ	GETUPC ;If it is, go print the UPC
SDA2	3A 4116	4481	ERROR1: LDA	SUCCESS ;Get the Success code
SDA5	FE 01	4482	CPI	CRD_CODE ;Look for CRD Error Code
SDA7	CA 4C32	4483	JZ	IDLE ;Go immediately to the Idle Loop if it is.
SDAA	AF	4484	XRA	A ;Clear A
SDAB	3C	4485	INR	A ;Make sure the Z bit is cleared
SDAC	2A 7219	4486	LHLD	MRKBUF ;Get the SP value
SDAF	F9	4487	SPHL	;Put it in the SP
SDB0	C9	4488	RET	;Leave
		4489		;
		4490		;
		4491		
		4492		
		4493		
		4494		;
		4495		;
SDB1	AF	4496	MARKSP: XRA	A ;Make a zero
SDB2	32 4116	4497	STA	SUCCESS ;Zero out this byte
SDB5	21 0002	4498	LXI	H,2 ;Offset for adding
SDB8	39	4499	DAD	SP ;Get the SP value and add it to 2
SDB9	22 7219	4500	SHLD	MRKBUF ;Save this value
SDBC	C9	4501	RET	;Leave
		4502		;
		4503		;
		4504		
		4505		
		4506		;
		4507		;
SDBD	7E	4508	HEXASC: MOV	A,M ;Get the digit to be converted
SDBE	E6 F0	4509	ANI	0F0H ;Save the high nibble
SDC0	1F	4510	RAR	;Move it down to the low nibble
SDC1	1F	4511	RAR	;
SDC2	1F	4512	RAR	;
SDC3	1F	4513	RAR	;
SDC4	CD SDD7	4514	CALL	CHANGE ;This actually does the convert
SDC7	12	4515	STAX	D ;Store the ASCII character
SDC8	13	4516	INX	D ;Point the Output buffer to the next byte
SDC9	7E	4517	MOV	A,M ;Get the Hex digit
SDCA	E6 0F	4518	ANI	0FH ;Save the low nibble
SDCC	CD SDD7	4519	CALL	CHANGE ;Go convert
SDCF	12	4520	STAX	D ;Put the ASCII character in the buffer
SDD0	13	4521	INX	D ;Bump the Output buffer pointer
SDD1	2B	4522	DCX	H ;Decrement the source pointer
SDD2	05	4523	DCR	B ;Decrement the count
SDD3	C2 SDBD	4524	JNZ	HEXASC ;If not done, loop
SDD6	C9	4525	RET	;Else, return
		4526		
SDD7	FE 0A	4527	CHANGE: CPI	0AH ;Check the value of the Hex digit
SDD9	DA SDE2	4528	JC	SKIP40 ;If less than A(Hex), jump

Error Addr	Code	Seq	Source statement
SDE1C	3D	4529	DCR A ;This will make the lower 3 bits correct
SDDD	E6 07	4530	ANI 07H ;Save them
SDDF	F6 40	4531	ORI 040H ;This completes the conversion
SDE1	C9	4532	RET ;Leave
SDE2	F6 30	4533	SKIP40: ORI 030H ;For values less than A(Hex), this is all
SDE4	C9	4534	RET ;Leave
		4535	;
		4536	;
		4537	
		4538	
		4539	
		4540	
		4541	;
		4542	;
SDE5	3A 40D3	4543	ACFAIL: LDA RUNFLG ;Get the runflag
SDE8	A7	4544	ANA A ;Is it set?
SDE9	CA 5DFD	4545	JZ ACFA10 ;Jump if not
SDEC	3E 01	4546	MVI A,01 ;Make a one
SDEE	32 40B5	4547	STA PFFLAG ;Set the Power Fail Flag
SDF1	D3 AC	4548	OUT SETPFI ;Set the Power Fail Interrupt signal
SDF3	D3 3B	4549	OUT SETBLK ;Set the 100Hz clock block
SDF5	3E 64	4550	MVI A,LARM3 ;Get the code for Loading and Arming counter 3
SDF7	D3 1D	4551	OUT ITCSR ;Send it out
SDF9	D3 3A	4552	OUT CLRBLK ;Clear the 100Hz clock block
SDFB	F1	4553	POP PSW ;Get the AC and the PSW back
SDFC	C9	4554	RET ;Go back to program mode
		4555	
SDFD	D3 3B	4556	ACFA10: OUT SETBLK ;Set the 100Hz clock block
SDFE	3E 64	4557	MVI A,LARM3 ;Get the code for Loading and Arming counter 3
SE01	D3 1D	4558	OUT ITCSR ;Send it out
SE03	D3 3A	4559	OUT CLRBLK ;Clear the 100Hz clock block
SE05	DB 20	4560	ACFA11: IN SUMREG ;Check the 2 ms timer
SE07	E6 40	4561	ANI 40H ;Look for overflow
SE09	C2 5E05	4562	JNZ ACFA11 ;Loop until the 2 ms done
SE0C	D3 2C	4563	ACFA12: OUT CLRTMR ;Turn off the Interval Timer
SE0E	20	4564	RIM ;Get the interrupt masks
SE0F	F6 09	4565	ORI 9H ;Set MSE and MS.5
SE11	30	4566	SIM ;Write the masks out
SE12	D3 3B	4567	OUT SETBLK ;Set the 100Hz clock block
SE14	3E C4	4568	MVI A,0C4H ;Disarm counter #3
SE16	D3 1D	4569	OUT ITCSR ;Send it out
SE18	3E E3	4570	MVI A,0E3H ;Code for setting output of counter #3 high
SE1A	D3 1D	4571	OUT ITCSR ;Send it out to the chip
SE1C	D3 3A	4572	OUT CLRBLK ;Unblock the 100Hz clock
SE1E	AF	4573	XRA A ;Make a 0
SE1F	32 40D3	4574	STA RUNFLG ;Zero the Run Flag
SE22	32 40B5	4575	STA PFFLAG ;Zero this flag
SE25	D3 AD	4576	OUT CLRPIFI ;Turn off this signal
SE27	D3 3C	4577	OUT CLRLIT ;Turn off the light
SE29	D3 20	4578	OUT CLRCLK ;Turn off the CPU clock

Error	Addr	Code	Seq	Source statement	
SE2B	D3 A7		4579	OUT DISMEM	;Turn off the CPU memory
SE2D	CD 4151		4580	CALL PUVEC	;Go perform the Power Up sequence
SE30	21 0000		4581	LXI H,0	;Make a word of 0
SE33	2B		4582	DCX H	;Start counting down
SE34	7C		4583	MOV A,H	;Get the high byte
SE35	B5		4584	ORA L	;OR in the low
SE36	C2 SE33		4585	JNZ ACFA13	;Jump if not 0
SE39	3E 10		4586	MVI A,10H	;Code for resetting RST7.5 if set
SE3B	30		4587	SIM	;Do it
SE3C	FB		4588	EI	;Turn on the interrupt system
SE3D	CD SE9E		4589	CALL MAKCON	;Go build the LS constants
SE40	DB 48		4590	IN TTDB	;Clear any garbage out of the Terminal DB
			4591		;and clear the RCV DONE bit as well
SE42	21 7341		4592	LXI H,PWRMSG	;Point to the Power Restored Message
SE45	F7		4593	RST 6	;Go send it
SE46	3A 707F		4594	LDA HMPEND	;Get this flag
SE49	1F		4595	RAR	;Were we trying to print a halt message?
SE4A	D8		4596	RC	;Return if we were
SE4B	3E 03		4597	MVI A,3	;Get the Power Fail code
SE4D	32 71DB		4598	STA HLTCOD	;Put it in the halt code
SE50	3E 0D		4599	MVI A,CR	;Get a carriage return
SE52	DF		4600	RST 3	;Go send it out
SE53	C3 58CA		4601	JMP CONHL1	;Now go try to start up the machine
			4602	:	
			4603	:	
			4604		
			4605		
			4606		
			4607	:	
			4608	:	
SE56	F5		4609	PARITY: PUSH PSW	;Save state
SE57	C5		4610	PUSH B	:
SE58	D5		4611	PUSH D	:
SE59	E5		4612	PUSH H	:
SE5A	D3 20		4613	OUT CLRCLK	;Turn off the clock
SE5C	AF		4614	XRA A	;Make a zero
SE5D	32 4196		4615	STA CLKFLG	;Clear the clock running flag
SE60	3A 40D3		4616	LDA RUNFLG	;Get the Run Flag
SE63	32 7222		4617	STA BPHALT	;Set this flag
SE66	D3 A7		4618	OUT DISMEM	;Turn off memory
SE68	CD 4993		4619	CALL WRTNOP	;this will get the current CSR into the NOP
			4620		;buffer
SE6B	3A 41A9		4621	LDA NOPBUF+2	;Get the MS byte
SE6E	17		4622	RAL	;Now get the MSB of the CSR into the carry
SE6F	3F		4623	CMC	;Flip it
SE70	1F		4624	RAR	;Everything is back where it belongs
SE71	32 41A9		4625	STA NOPBUF+2	;Store it away
SE74	CD 49A7		4626	CALL LDCSR1	;Go restore the CSR
SE77	D3 A6		4627	OUT ENBMEM	;Turn memory back on
SE79	3E 0D		4628	MVI A,CR	;Put a Carriage Return in A

Error	Addr	Code	Seq	Source statement
SE7B	DF		4629	RST 3 ;Go print it
SE7C	CD 59B6		4630	CALL GETUPC ;Go get and print the UPC
SE7F	E1		4631	POP H ;Restore state
SE80	D1		4632	POP D ;
SE81	C1		4633	POP B ;
SE82	3A 721F		4634	LDA MICFLG ;Check the Micro Step Flag
SE85	1F		4635	RAR ;Is it set?
SE86	D2 SE8C		4636	JNC 1\$;Jump if not
SE89	F1		4637	POP PSW ;Finish restoring state
SE8A	FB		4638	EI ;Enable interrupts
SE8B	C9		4639	RET ;and leave
SE8C	AF		4640	1\$: XRA A ;Make a zero
SE8D	32 40D3		4641	STA RUNFLG ;Zero the run flag
SE90	D3 3C		4642	OUT CLRLIT ;Turn off the run light
SE92	F1		4643	POP PSW ;Restore the PSW
SE93	22 4168		4644	SHLD TMPBUF ;Save HL for a bit
SE96	E1		4645	POP H ;Fix up the Stack
SE97	2A 4168		4646	LHLD TMPBUF ;Restore HL
SE9A	FB		4647	EI ;Turn on interrupts again
SE9B	C3 4C32		4648	JMP IDLE ;Go to console mode
			4649	;
			4650	;
			4651	;
			4652	;
			4653	;
			4654	;
			4655	;
SE9E	D3 20		4656	MAKCON: OUT CLRCLK ;Turn off the clock
SEA0	D3 A1		4657	OUT DISPAR ;Turn off parity
SEA2	21 4197		4658	LXI H,PARFLG ;Point to the Parity flag
SEA5	7E		4659	MOV A,M ;Get it into A
SEA6	36 00		4660	MVI M,0 ;Now zero it
SEA8	23		4661	INX H ;Bump the pointer to a temp
SEA9	77		4662	MOV M,A ;Save this copy
SEAA	D3 A7		4663	OUT DISMEM ;Turn off memory
SEAC	21 74A9		4664	LXI H,BASCON ;Point to a work area
SEAF	11 7186		4665	LXI D,POS1 ;Point to a positive one
SEB2	CD 00F9		4666	CALL COPY ;Copy it over
SEB5	21 74AD		4667	LXI H,CONBUF ;Point to the storage area for the constants
SEB8	7E		4668	MAKCN1: MOV A,M ;Get the count for the number of loads in the
			4669	group
SEB9	A7		4670	ANA A ;Check for zero
SEBA	CA 5F16		4671	JZ MAKCN4 ;Jump over next part if it is
SEBD	32 74A6		4672	STA GRPCNT ;Save it
SEC0	23		4673	INX H ;Bump the pointer
SEC1	7E		4674	MOV A,M ;Get the Base address for this group
SEC2	32 74A7		4675	STA BASADR ;Save it
SEC5	23		4676	INX H ;Bump the pointer again
SEC6	7E		4677	MOV A,M ;Get the a count for mapping the bytes in the
			4678	buffer into the LS load word

Error	Addr	Code	Seq	Source statement	
SEC7	32 74A8	4679		STA CONS12	;Save it
SECA	23	4680		INX H	;Bump the pointer
SECB	E5	4681		PUSH H	;Save the pointer value
SECC	21 7472	4682	MAKCN2:	LXI H,SHFBUF	;Point to the shift buffer
SECF	CD 6AFA	4683		CALL ZROBUF	;Go zero it
SED2	D1	4684		POP D	;Get the pointer back
SED3	3A 74A8	4685		LDA CONS12	;Get the number of bytes per entry
SED6	47	4686		MOV B,A	;Put it in B
SED7	FE 05	4687		CPI 05	;Check for count of five
SED9	CA 5F39	4688		JZ MAKCN6	;Jump if it is
SEDC	1A	4689	MAKCN3:	LDAX D	;Get a byte
SEDD	77	4690		MOV M,A	;Store it
SEDE	23	4691		INX H	;Bump the pointer in HL
SEDF	13	4692		INX D	;and the one in DE
SEE0	05	4693		DCR B	;Reduce the entry count by one
SEE1	C2 SEDC	4694		JNZ MAKCN3	;Loop back if not zero
SEE4	75	4695		PUSH D	;Save the current pointer value
SEES	CD 5F4D	4696	MAKCN8:	CALL FASTWR	;Go write WR 0 the quick way
SEEB	21 7472	4697		LXI H,SHFBUF	;Point the shift buffer
SECE	11 747D	4698		LXI D,WRNLSN	;Point to the LSn to WRn U word
SECF	06 03	4699		MVI B,03	;Set up the count
SEFU	CD 00FB	4700		CALL COPY1	;Go copy them
SEF3	2B	4701		DCX H	;Move the pointer back
SEF4	2B	4702		DCX H	;to point to byte 2
SEF5	3A 74A7	4703		LDA BASADR	;Get the address
SEF8	77	4704		MOV M,A	;Put it in the second byte
SEF9	3C	4705		INR A	;Bump the address
SEFA	32 74A7	4706		STA BASADR	;Save the new value
SEFD	2B	4707		DCX H	;Point to byte one
SEFE	E5	4708		PUSH H	;Save this address
SEFF	0E 03	4709		MVI C,03	;Number of bytes to shift accross
SF01	CD 4A00	4710		CALL LSHIFT	;Position the U word correctly
SF04	D1	4711		POP D	;Get the address back
SF05	CD 5649	4712		CALL EXECUT	;Go do it
SF08	3A 74A6	4713		LDA GRPCNT	;Get the count
SF0B	3D	4714		DCR A	;Subtract one
SF0C	32 74A6	4715		STA GRPCNT	;Save the new value
SF0F	C2 SECC	4716		JNZ MAKCN2	;Loop back if not done
SF12	E1	4717		POP H	;Put the pointer in HL
SF13	C3 SEB8	4718		JMP MAKCN1	;Go here if this group done
		4719			
SF16	CD 4993	4720	MAKCN4:	CALL WRTNOP	;Put a NOP in the CSR
SF19	21 41AB	4721		LXI H,SWAP+1	;Point to high byte of Swap buffer
SF1C	AF	4722		XRA A	;Make a zero
SF1D	77	4723		MOV M,A	;Send it out
SF1E	2B	4724		DCX H	;Drop the pointer down one
SF1F	77	4725		MOV M,A	;Zero the low byte
SF20	CD 49D1	4726		CALL LDUPC	;Go load it in
SF23	D3 A6	4727		OUT ENBMEM	;Turn on memory again
SF25	21 4198	4728		LXI H,PARTMP	;Point to the Parity Flag copy

Error Addr	Code	Seq	Source statement	
SF28	7E	4729	MOV A,M	;Get the copy
SF29	2B	4730	DCX H	;Point to the Parity Flag byte
SF2A	77	4731	MOV M,A	;Send the copy out
SF2B	1F	4732	RAR	;See if we should turn parity back on
SF2C	D2 SF31	4733	JNC MAKCNS	;Jump if not
SF2F	D3 A0	4734	OUT ENBPARG	;Turn Parity on again
SF31	3A 4196	4735	MAKCNS: LDA CLKFLG	;Get the clock flag
SF34	1F	4736	RAR	;Check it
SF35	D0	4737	RNC	;Done if not set
SF36	D3 21	4738	OUT SETCLK	;Turn on the clock
SF38	C9	4739	RET	;Leave
		4740		
SF39	D5	4741	MAKCNS: PUSH D	;Save DE
SF3A	11 74A9	4742	LXI D,BASCON	;Point to the constant
SF3D	D5	4743	PUSH D	;Save the address
SF3E	21 7472	4744	LXI H,SHFBUF	;Point to the destination
SF41	CD 00F9	4745	CALL COPY	;Go copy the data over
SF44	E1	4746	POP H	;Get the address back
SF45	0E 04	4747	MVI C,04	;Set the count
SF47	CD 4A00	4748	CALL LSHIFT	;Go shift 4 bytes left by one bit
SF4A	C3 5EE5	4749	JMP MAKCN8	;Return to the main flow
		4750		
		4751		
SF4D	CD 4993	4752	FASTWR: CALL WRTNOP	;NOP the CSR
SF50	21 0016	4753	LXI H,FASUPC	;Load up the address of the right routine
SF53	22 41AA	4754	SHLD SWAP	;Put it in the swapping buffer
SF56	21 41AA	4755	LXI H,SWAP	;Point to it
SF59	CD 49D1	4756	CALL LDUPC	;Go load it
SF5C	3E 21	4757	MVI A,33	;Loop control count
SF5E	32 40B4	4758	STA TEMP	;Put it here
SF61	D3 23	4759	OUT SETSS	;Do the NOP
SF63	D3 22	4760	OUT CLRSS	
SF65	D3 23	4761	FASWR1: OUT SETSS	;First tick in a three tick loop
SF67	D3 22	4762	OUT CLRSS	
SF69	D3 AB	4763	OUT CLRATN	;Clear Console attention
SF6B	21 7472	4764	LXI H,SHFBUF	;Point to the data
SF6E	0E 04	4765	MVI C,04	;Number of bytes to shift accross
SF70	CD 4A00	4766	CALL LSHIFT	;Go shift it left
SF73	D2 SF78	4767	JNC FASWR2	;Jump if no carry
SF76	D3 AA	4768	OUT SETATN	;Set Console Attention
SF78	D3 23	4769	FASWR2: OUT SETSS	;Second tick
SF7A	D3 22	4770	OUT CLRSS	
SF7C	D3 23	4771	OUT SETSS	;Third tick
SF7E	D3 22	4772	OUT CLRSS	
SF80	3A 40B4	4773	LDA TEMP	;Get the count
SF83	3D	4774	DCR A	;Subtract one
SF84	32 40B4	4775	STA TEMP	;Save it
SF87	C2 SF65	4776	JNZ FASWR1	;Loop back if not done
SF8A	C9	4777	RET	;Leave
		4778		

Error	Addr	Code	Seq	Source statement
			4779	;*****
			4780	
			4781	
			4782	
			4783	;*****
			4784	
SF8B	3A	40B6	4785	PUBCHK: LDA PUBFLG ;Point to the power up boot flag
SF8E	A7		4786	ANA A ;Is it zero?
SF8F	C2	4C32	4787	JNZ IDLE ;Jump if not
SF92	21	7323	4788	LXI H,PUBMSG ;Point to the Power Up Boot message
SF95	F7		4789	RST 6 ;Go send it
SF96	C3	0040	4790	JMP INIVEC ;Then go down into the ROM
			4791	
			4792	;*****
			4793	
			4794	
			4795	;*****
			4796	
SF99			4797	COLD_BEGIN:
SF99	D3	A7	4798	OUT DISMEM ;Turn off memory
SF9B	AF		4799	XRA A ;Make a 0
SF9C	32	4196	4800	STA CLKFLG ;Zero this flag
SF9F	32	408A	4801	STA COLD ;and this one
SFA2	3C		4802	INR A ;Make a one
SFA3	32	40B6	4803	STA PUBFLG ;Set the Power Up Boot in Progress Flag
SFA6	32	4197	4804	STA PARFLG ;Zero this flag
SFA9	CD	5E9E	4805	CALL MAKCON ;Go make the constants
SFAC	20		4806	RIM ;Read in the interrupt masks
SFAD	F6	08	4807	ORI 08H ;Set the MSE
SFAF	E6	0D	4808	ANI 0DH ;Clear the Parity interrupt mask
SFB1	30		4809	SIM ;Set the masks
SFB2	FB		4810	EI ;Enable interrupts
SFB3	CD	6999	4811	CALL THRRPR ;Go fix up the timers
SFB6	3A	4150	4812	LDA DEFDRV ;Get the Default Drive number
SFB9	32	4091	4813	STA UNIT ;Set up the unit number
SFBC	3E	01	4814	MVI A,1 ;Make a 1
SFBE	32	409D	4815	STA INDFLG ;Set this flag
SFC1	21	7239	4816	LXI H,PWRFIL ;Point to this command
SFC4	22	721C	4817	SHLD INDADR ;Save the pointer
SFC7	CD	4C5E	4818	CALL CNSOLB ;Go process it
SFCA	C2	4C32	4819	JNZ IDLE ;Stop if we return with Z bit not set
SFCD	3E	01	4820	MVI A,1 ;Make a 1
SFCF	32	40D5	4821	STA WCS_LOAD_FLAGS ;Set the POWER.CMD done flag
			4822	
SFD2			4823	COLD_BEGIN0:
SFD2	3E	01	4824	MVI A,1 ;Make a 1
SFD4	32	409D	4825	STA INDFLG ;Set this flag
SFD7	21	7247	4826	LXI H,CODFIL ;Point to this command
SFDA	22	721C	4827	SHLD INDADR ;Save the pointer
SFDD	CD	4C5E	4828	CALL CNSOLB ;Go process it

Error	Addr	Code	Seq	Source statement	
SFE0	C2 4C32	4829	JNZ	IDLE	;Jump if Z bit clear
SFE3	3E 03	4830	MVI	A,3	;Make a 1
SFE5	32 40D5	4831	STA	WCS_LOAD_FLAGS	;Set the CODE0n.CMD done flag.
		4832			
SFE8		4833		COLD_BEGIN1:	
SFE8	21 40FD	4834	LXI	H,CRD_FLAGS	;Point to the CRD flags
SFEB	7E	4835	MOV	A,M	;Get them
SFEC	36 00	4836	MVI	M,0	;Zero out the copy in Memory
SFEE	1F	4837	RAR		;Check for CRD in Progress
SFEF	D2 601A	4838	JNC	COLD_HALT_TEST	;Jump if not.
SFF2	1F	4839	RAR		;Check the Micros Completed flag
SFF3	DC 5D78	4840	CC	CRD_ERROR	;Go print an error message.
SFF6	1F	4841	RAR		;Check for Micro Errors.
SFF7	DA 6010	4842	JC	2\$	;Jump if errors
SFFA	1F	4843	RAR		;Check for Auto or Menu Mode.
SFFB	21 727C	4844	LXI	H,CRAFIL+3	;Point to the Auto Mode File String.
SFFE	D2 6004	4845	JNC	1\$	;Jump if Auto Mode
6001	21 728E	4846	LXI	H,CRMFIL+3	;Otherwise, point to the Menu Mode Boot file.
6004	3A 4150	4847	LDA	DEFDRV	;Get the Default Drive Number.
6007	F6 30	4848	ORI	30H	;Make it ASCII.
6009	77	4849	MOV	M,A	;Store it in the string.
600A	2B	4850	DCX	H	;Back the pointer up to the beginning of
600B	2B	4851	DCX	H	;the string.
600C	2B	4852	DCX	H	;
600D	C3 602F	4853	JMP	CLDST1	;Go boot things up.
6010	AF	4854	2\$: XRA	A	;Make a 0
6011	32 40BB	4855	STA	OFLAG	;Clear the Control 0 Flag
6014	3E 0D	4856	MVI	A,CR	;Put a Carriage Return in A
6016	DF	4857	RST	3	;Go send it out
6017	C3 4C32	4858	JMP	IDLE	;Don't Boot Up if the Micros had errors.
		4859			
601A		4860		COLD_HALT_TEST:	
601A	DB 02	4861	IN	HLTFLG	;Check the Halt/Auto Restart Switch
601C	1F	4862	RAR		;Put it in the carry
601D	D2 4C32	4863	JNC	IDLE	;Go to the Idle Loop if low
		4864			
6020	21 72EB	4865	CLDSTX: LXI	H,CLDMSG	;Point to this message
6023	F7	4866	RST	6	;Go print it
6024	3A 4150	4867	CLDSTR: LDA	DEFDRV	;Get the default drive number
6027	F6 30	4868	CLDST0: ORI	30H	;Make it ASCII
6029	32 7258	4869	STA	DEFFIL+3	;Put it into the command string
602C	21 7255	4870	LXI	H,DEFFIL	;Point to this command string
602F	22 721C	4871	CLDST1: SHLD	INDADR	;Save the address of the indirect command file
6032	21 408A	4872	LXI	H,COLD	;Point to this flag
6035	7E	4873	MOV	A,M	;Get it
6036	A7	4874	ANA	A	;Is it zero?
6037	C4 6062	4875	CNZ	CFAIL	;Jump if not
603A	3E 01	4876	MVI	A,1	;Force a one into A
603C	32 408A	4877	STA	COLD	;Set this flag
603F	32 41B7	4878	STA	WARM	;and this flag also

Error	Addr	Code	Seq	Source statement	
6042	32	409D	4879	STA INDFLG	;Set this flag
6045	CD	56F8	4880	CALL INISUB	;Go init the machine
6048	C4	6062	4881	CNZ CFAIL	;If error try a Power up Boot
604B	21	71B6	4882	LXI H,EDPACK	;Point to this packet area
604E	11	7498	4883	LXI D,SPPACK	;Point to this packet
6051	06	0A	4884	MVI B,10	;Set up the count
6053	CD	00FB	4885	CALL COPY1	;Go copy it over
6056	CD	6222	4886	CALL SNDRCV	;Go send and receive a packet
6059	C4	6062	4887	CNZ CFAIL	;If error, go check the PUB flag
605C	CD	4C5E	4888	CALL CNSOLB	;Now go process the indirect command file we
			4889		;set up at CLDSTR
605F	CD	6062	4890	CALL CFAIL	;If we come back here, there is an error
			4891		
6062	E1		4892	CFAIL: POP H	;Get the error PC
6063	22	7565	4893	SHLD CFATAL	;Store it
6066	21	730A	4894	LXI H,CFMSG	;Point to the Cold Failed message
6069	F7		4895	RST 6	;Go print it
606A	3A	4116	4896	LDA SUCCES	;Get the success code
606D	A7		4897	ANA A	;If we had a tape error, this will be non zero
606E	CA	5F8B	4898	JZ PUBCHK	;Go see about doing a Power Up Boot
6071	C3	4C32	4899	JMP IDLE	;Otherwise, go to the Idle Loop
			4900		
			4901		;*****
			4902		
			4903		
			4904		;*****
			4905		
6074	31	4080	4906	WRMSTR: LXI SP,STACK	;Set up the stack pointer
6077	21	72AD	4907	LXI H,WRMMSG	;Go send the Attempting Warm Restart message
607A	F7		4908	RST 6	;Do it
607B	AF		4909	XRA A	;Make a 0
607C	32	7564	4910	STA WEFLAG	;Zero out this flag
607F	3A	41B7	4911	LDA WARM	;Get this flag
6082	A7		4912	ANA A	;See if it is zero
6083	C4	61E3	4913	CNZ WFAIL	;Jump if not
6086	CD	56F8	4914	CALL INISUB	;Go init the machine
6089	C4	61E3	4915	CNZ WFAIL	;If error at this point, go try cold start
608C	21	71CC	4916	LXI H,SCNADR	;Point to the address part of this packet
608F	11	71A6	4917	LXI D,NEG200	;Point to a negative 200 hex
6092	CD	00F9	4918	CALL COPY	;Go copy it over
6095	3E	01	4919	MVI A,1	;Make a one
6097	32	40D4	4920	STA NOTYPE	;Suppress error typeouts
609A	E7		4921	WRMST1: RST 4	;Do this to allow for breakouts
609B	21	71CC	4922	LXI H,SCNADR	;Point to the address part of SCNPAK
609E	01	7192	4923	LXI B,POS200	;Point to a positive 200 hex
60A1	CD	6251	4924	CALL ADDAD1	;Go add it in
60A4	21	71B6	4925	LXI H,EDPACK	;Now we move SCNPAK to EDPACK
60A7	11	71CA	4926	LXI D,SCNPAK	;Point to the source
60AA	06	06	4927	MVI B,6	;Set up the count
60AC	CD	00FB	4928	CALL COPY1	;Copy source to dest

Error Addr	Code	Seq	Source statement
60AF	CD 6204	4929	CALL SNDPRP ;Go set up the send and receive packets
60B2	C4 61F2	4930	CNZ WFAIL1 ;Loop back if error
60B5	CD 6233	4931	CALL DATCHK ;See if data is same as address
60B8	C4 61F2	4932	CNZ WFAIL1 ;Jump if not the same
60BB	CD 6204	4933	CALL SNDPRP ;Go send and receive the packets
60BE	C4 61F2	4934	CNZ WFAIL1 ;Jump if error occurred
60C1	CD 6233	4935	CALL DATCHK ;See if address is same as data
60C4	CC 61F2	4936	CZ WFAIL1 ;Jump if they are (shouldn't be)
60C7	21 71E4	4937	LXI H,RESADR ;Point to this storage location
60CA	11 71C6	4938	LXI D,RCV DAT ;and the the data to go there
60CD	CD 00F9	4939	CALL COPY ;Go copy it over
60D0	CD 6204	4940	CALL SNDPRP ;Go get the next longword
60D3	C4 61F2	4941	CNZ WFAIL1 ;Jump if error
60D6	21 71E8	4942	LXI H,LNGSAV ;Point to this storage
60D9	11 71C6	4943	LXI D,RCV DAT ;and this data
60DC	CD 00F9	4944	CALL COPY ;Move it over
60DF	21 71EC	4945	LXI H,LNGSUM ;Point to this storage area
60E2	CD 6AFA	4946	CALL ZROBUF ;Go zero it out
60E5	CD 6204	4947	CALL SNDPRP ;Get the next longword
60E8	C4 61F2	4948	CNZ WFAIL1 ;Loop back if error
60EB	21 71F2	4949	LXI H,WFADR ;Point to the beginning of a storage area
60EE	11 71C2	4950	LXI D,RCVADR ;Point the beginning of the address part
60F1	06 08	4951	MVI B,8 ;Count for copying
60F3	CD 00FB	4952	CALL COPY1 ;Go copy
60F6	21 71FA	4953	LXI H,WRMCNT ;Point to this counter byte
60F9	36 1F	4954	MVI M,1FH ;Set up the count
60FB	21 71B8	4955	LXI H,EDADDR ;Point to the address part of the packet
60FE	11 71E4	4956	LXI D,RESADR ;Point to the restart address
6101	CD 00F9	4957	CALL COPY ;Go copy it over
6104	CD 6204	4958	WRMST2: CALL SNDPRP ;Get the next longword
6107	C4 61F2	4959	CNZ WFAIL1 ;Jump if error
610A	21 71EC	4960	LXI H,LNGSUM ;Point to the Longword checksum
610D	01 71C6	4961	LXI B,RCV DAT ;Point to the data
6110	CD 6251	4962	CALL ADDAD1 ;Go add it in
6113	21 71FA	4963	LXI H,WRMCNT ;Point to this counter
6116	35	4964	DCR M ;Decrement it
6117	C2 6104	4965	JNZ WRMST2 ;Loop until zero
611A	21 71EC	4966	LXI H,LNGSUM ;Point to the checksum
611D	11 71E8	4967	LXI D,LNGSAV ;Point to the one we got from the RPB
6120	06 04	4968	MVI B,4 ;Set up a count of 4
6122	1A	4969	WRMST9: LDAX D ;Get a byte
6123	BE	4970	CMP M ;See if equal
6124	C4 61F2	4971	CNZ WFAIL1 ;Jump back if not
6127	23	4972	INX H ;Bump the pointer
6128	13	4973	INX D ;and the other pointer
6129	05	4974	DCR B ;Reduce the count
612A	C2 6122	4975	JNZ WRMST9 ;Loop back until count equals 0
612D	AF	4976	XRA A ;Make a 0
612E	32 40D4	4977	STA NOTYPE ;Clear this flag
6131	21 71F6	4978	LXI H,WFDATA ;Point to the low byte

Error Addr	Code	Seq	Source statement	
6134	7E	4979	MOV A,M	;Get it
6135	1F	4980	RAR	;Put the LSB in the carry
6136	DC 61E3	4981	CC WFAIL	;Jump if set
6139	3F	4982	CMC	;Set it if it was clear
613A	17	4983	RAL	;Rotate things back
613B	77	4984	MOV M,A	;Send the low byte back
613C	21 71B6	4985	LXI H,EDPACK	;Point to this area
613F	11 71D0	4986	LXI D,DEPPAK	;Point to this source packet
6142	06 06	4987	MVI B,6	;Set up a count
6144	CD 00FB	4988	CALL COPY1	;Go copy it over
6147	21 71BC	4989	LXI H,EDDATA	;Point to the data part of the packet
614A	11 71DC	4990	LXI D,HALTPC	;Point to the Halt PC
614D	CD 00F9	4991	CALL COPY	;Put it in the data part of the packet
6150	CD 6222	4992	CALL SNDRCV	;Go send it
6153	C4 61E3	4993	CNZ WFAIL	;If error, go try a cold start
6156	21 71BC	4994	LXI H,EDDATA	;Point to the data part again
6159	11 71E0	4995	LXI D,HLTPSL	;Point to the Halt PSL
615C	CD 00F9	4996	CALL COPY	;Copy it over
615F	01 71B6	4997	LXI B,POS1	;Point to a 1
6162	CD 624E	4998	CALL ADDADR	;Go add it
6165	CD 6222	4999	CALL SNDRCV	;Go send the PSL to R11
6168	C4 61E3	5000	CNZ WFAIL	;Go try a cold start if error
616B	21 71BC	5001	LXI H,EDDATA	;Point to this area
616E	CD 6AFA	5002	CALL ZROBUF	;Go zero it
6171	3A 71DB	5003	LDA HLTCOD	;Get the halt code
6174	32 71BC	5004	STA EDDATA	;Put it in the packet
6177	01 71B6	5005	LXI B,POS1	;Now prepare to bump the address
617A	CD 624E	5006	CALL ADDADR	;Go do it
617D	CD 6222	5007	CALL SNDRCV	;Go send and receive
6180	C4 61E3	5008	CNZ WFAIL	;Go try a cold start if error
6183	3E 0E	5009	MVI A,0EH	;Get the address of the SP
6185	32 71B8	5010	STA EDADDR	;Put it into the address area of the packet
6188	21 71BC	5011	LXI H,EDDATA	;Point to the data area
618B	11 71CC	5012	LXI D,SCNADR	;Point to this address
618E	CD 00F9	5013	CALL COPY	;Put it into the packet
6191	21 71BC	5014	LXI H,EDDATA	;Point to this part of the packet again
6194	01 7192	5015	LXI B,POS200	;Positive 200
6197	CD 6251	5016	CALL ADDAD1	;Go add it in
619A	CD 6222	5017	CALL SNDRCV	;Go send it
619D	C4 61E3	5018	CNZ WFAIL	;Go try a cold start if error
61A0	01 71B6	5019	LXI B,POS1	;Positive 1
61A3	CD 624E	5020	CALL ADDADR	;Go bump the address part of the packet
61A6	21 71BC	5021	LXI H,EDDATA	;Point to the data portion
61A9	11 71E4	5022	LXI D,RESADR	;Point to the restart address
61AC	CD 00F9	5023	CALL COPY	;Go copy it over
61AF	CD 6222	5024	CALL SNDRCV	;Go send the packet
61B2	C4 61E3	5025	CNZ WFAIL	;Go try a cold start if error
61B5	21 71B6	5026	LXI H,EDPACK	;Point to this packet
61B8	11 71F0	5027	LXI D,WFPACK	;and this source packet
61BB	06 0A	5028	MVI B,10	;Count for copying

Error Addr	Code	Seq	Source statement	
61BD	CD 00FB	5029	CALL COPY1	;Go move the packet over
61C0	CD 6222	5030	CALL SNDRCV	;Go send and receive the deposit
61C3	C4 61E3	5031	CNZ WFAIL	;Go try a cold start if error
61C6	3E 01	5032	MVI A,1	;Make a one
61C8	32 41B7	5033	STA WARM	;Set the Console based Warm Flag
61CB	AF	5034	XRA A	;Make a 0
61CC	32 71FD	5035	STA CONMOD	;Zero this Continue Modifier byte
61CF	21 71FC	5036	LXI H,CONPAK	;Point to the continue packet
61D2	CD 622B	5037	CALL SNDRCX	;Go send it
61D5	C4 61E3	5038	CNZ WFAIL	;Go try a cold start if error
61D8	AF	5039	XRA A	;Make a 0
61D9	32 40D4	5040	STA NOTYPE	;Zero the No Error Timeout flag
61DC	3C	5041	INR A	;Make a 1
61DD	32 40D3	5042	STA RUNFLG	;Turn on the Run Flag
61E0	C3 5877	5043	JMP CNTIN2	;Go share code
		5044		
61E3	AF	5045	WFAIL: XRA A	;Make a 0
61E4	32 40D4	5046	STA NOTYPE	;Zero out this flag
61E7	E1	5047	POP H	;Get the error PC
61E8	22 7567	5048	SHLD WFATAL	;Store it here
61EB	21 72CF	5049	LXI H,WFMMSG	;Point to the Warm Failed message
61EE	F7	5050	RST 6	;Go print it
61EF	C3 6020	5051	JMP CLDSTX	;Go try a cold restart
		5052		
61F2	E1	5053	WFAIL1: POP H	;Get the failure PC
61F3	3A 7564	5054	LDA WEFLAG	;Get this flag
61F6	A7	5055	ANA A	;See if set
61F7	C2 609A	5056	JNZ WRMST1	;Jump if set
61FA	3C	5057	INR A	;Make a one
61FB	32 7564	5058	STA WEFLAG	;Set it
61FE	22 7569	5059	SHLD WERROR	;Save the error PC
6201	C3 609A	5060	JMP WRMST1	;Go back to try again
		5061		
		5062		
6204	21 74A2	5063	SNDPRP: LXI H,HEMSIZ	;Point to the highest memory address
6207	11 71B8	5064	LXI D,EDADDR	;Point to next memory to be accessed
620A	CD 6242	5065	CALL CMPCHK	;Go compare them
620D	C2 6214	5066	JNZ SNDPR1	;Jump if not the same
6210	E1	5067	POP H	;Error if the same, so fix stack
6211	C3 6024	5068	JMP CLDSTR	;and go try a cold start
6214	CD 6222	5069	SNDPR1: CALL SNDRCV	;Go send and receive packets
6217	01 718E	5070	LXI B,POS4	;Point to a positive 4
621A	CD 624E	5071	CALL ADDADR	;Go add 4 to EDADDR
621D	3A 4116	5072	LDA SUCCES	;Get the success code
6220	A7	5073	ANA A	;Set or clear the Z bit
6221	C9	5074	RET	;Leave
		5075		
		5076		
6222	CD 5DB1	5077	SNDRCV: CALL MARKSP	;Go mark the SP value
6225	CD 514B	5078	CALL SNDMSX	;Go send it

Error	Addr	Code	Seq	Source statement	
6220	C3	5186	5079	JMP	RCVMSX ;Get the reply
			5080		
622B	EB		5081	SNDRCX: XCHG	;Put the address in DE for a bit
622C	CD	5DB1	5082	CALL	MARKSP ;Go mark the SP value
622F	EB		5083	XCHG	;Get the address back
6230	C3	514E	5084	JMP	SNDMSG ;Go send it
			5085		
6233	21	71CC	5086	DATCHK: LXI	H,SCNADR ;Point to the address part of this packet
6236	11	71C6	5087	LXI	D,RCVDAT ;and the data part of this one
6239	C3	6242	5088	JMP	CMPCHK ;Go compare them
623C	21	418E	5089	XFRCHK: LXI	H,XFRCNT ;Point to the Transfer count
623F	11	7182	5090	ZROCHK: LXI	D,ZERO ;Point to a longword zero
6242	06	04	5091	CMPCHK: MVI	B,4 ;Set up the count in B
6244	1A		5092	CMPCH1: LDAX	D ;Get the low byte of the source
6245	BE		5093	CMP	H ;Compare it to a byte of the dest
6246	C0		5094	RNZ	;Leave if not the same
6247	23		5095	INX	H ;Bump the dest pointer
6248	13		5096	INX	D ;and the source pointer
6249	05		5097	DCR	B ;Decrement the count
624A	C2	6244	5098	JNZ	CMPCH1 ;Keep looping until count=0
624D	C9		5099	RET	;Leave with Z bit set
			5100		
624E	21	71B8	5101	ADDADR: LXI	H,EDADDR ;Point to the address buffer
6251	1E	04	5102	ADDAD1: MVI	E,04H ;Prep the loop count
6253	AF		5103	^RA	A ;Clear the carry
6254	0A		5104	1\$: LDAX	B ;Get the low byte of the constant
6255	8E		5105	ADC	H ;Add it to the address
6256	77		5106	MOV	H,A ;Store it
6257	03		5107	INX	B ;Bump the constant pointer
6258	23		5108	INX	H ;Bump the address pointer
6259	1D		5109	DCR	E ;Decrement the loop count
625A	C2	6254	5110	JNZ	1\$;If not zero, repeat the loop
625D	AF		5111	XRA	A ;Clear the accumulator.
625E	C9		5112	RET	;Leave.
			5113		
			5114	:	
			5115	;	
			5116	:	
			5117	;	CONSOLE MODE ENDS HERE.
			5118	:	
			5119	;	
			5120		
			5121		
			5122		
			5123		
			5124		
			5125		
			5126	;	
			5127	:	
			5128	;	PROGRAM MODE BEGINS HERE.

Error Addr	Code	Seq	Source statement
		5129	:
		5130	:.....
		5131	:
		5132	:
		5133	:
		5134	: IPR=
		5135	:
		5136	:07 06 05 04 03 02 01 00
		5137	:-----
		5138	:PRI1 PRI2 PRI3 PRI4 N.U. N.U. N.U. N.U.
		5139	:-----
		5140	:
		5141	:
		5142	: ISR=
		5143	:
		5144	:
		5145	:-----
		5146	:TTR TTR TTX TTX TUX TUX TUR TUR
		5147	:I.S. I.E. I.S. I.E. I.E. I.S. I.E. I.S.
		5148	:-----
		5149	:
		5150	:
		5151	:
		5152	:
		5153	: ICCS=
		5154	:
		5155	:07 06 05 04 03 02 01 00
		5156	:-----
		5157	:I.E INT ERR N.U. N.U. SGL XFR RUN
		5158	:-----
		5159	:
		5160	:
		5161	:
		5162	:
		5163	: THIS IS THE BEGINNING OF THE PROGRAM I/O MODE CODE
		5164	:
		5165	:PROGRAM I/O MODE consists of a primary loop that handles the peripheral
		5166	:devices and periodically checks for CPU service requests, routines to
		5167	:service MFPR's, MTPR's and interrupt Acknowledges, and routines to handle the
		5168	:maintenance of the Interval Timer
		5169	:To keep latency reasonable there are a number of CPU service request checks.
		5170	:
		5171	:
		5172	:
625F	D3 3D	5173	ENTER: OUT SETLIT ;Turn on the Run light
6261	AF	5174	XRA A ;Make a 0
6262	32 40BB	5175	STA OFLAG ;Clear the Control 0 flag
6265	32 40CD	5176	STA SFLAG ;and the Control S flag
6268	32 409D	5177	STA INDFLG ;Clear the Indirect Command Flag
626B	32 70FC	5178	STA PCOUNT ;Zero the Control P Timeout

Error	Addr	Code	Seq	Source statement
626E	3C		5179	INR A ;Make a 1
626F	32 7098		5180	STA BLKFLG ;Set the 100Hz clock blocked flag
6272	D3 3B		5181	OUT SETBLK ;Now block it
6274	3E C4		5182	MVI A,0C4H ;Get the code to disarm counter #3
6276	D3 1D		5183	OUT ITCR ;Send it out
6278	3E E3		5184	MVI A,0E3H ;Code for setting output of counter #3 high
627A	D3 1D		5185	OUT ITCR ;Send it out to the chip
627C	AF		5186	XRA A ;Make a 0
627D	32 7098		5187	STA BLKFLG ;Clear the 100Hz clock blocked flag
6280	D3 3A		5188	OUT CLRBLK ;Now clear the block
6282	21 707C		5189	LXI H,ICCS ;Point to the ICCS
6285	7E		5190	MOV A,M ;Get it
6286	1F		5191	RAR ;Check the run bit
6287	D2 6296		5192	JNC 1\$;Jump if not set
628A	17		5193	RAL ;Move things back into position
628B	F6 20		5194	ORI 20H ;Set the error bit
628D	77		5195	MOV M,A ;Put it back
628E	20		5196	RIM ;Get the interrupt masks
628F	E6 FE		5197	ANI 0FEH ;Clear MS.5
6291	F6 08		5198	ORI 08H ;Set MSE
6293	30		5199	SIM ;Write the masks
6294	D3 2D		5200	OUT SETTMR ;Turn on the Interval Timer
6296	AF		5201	XRA A ;Make a zero again
6297	32 40F4		5202	STA SLOW_CLK_FLG ;Clear out this flag.
629A	CD 66B6		5203	CALL IPRCHK ;Go see about setting Console Attention
			5204	
629D	3A 40D7		5205	CPCHK1: LDA SPEND ;Get the ^S Pending/Sent/Taken flags
62A0	E6 04		5206	ANI 4 ;Save only the Taken flag
62A2	CA 62BD		5207	JZ CPCHO ;Jump if not set
62A5	3A 7E06		5208	LDA TRNFLG ;Get the Trans Mode flag.
62A8	A7		5209	ANA A ;Are we in Protocol or Trans mode.
62A9	C2 62BD		5210	JNZ CPCHO ;IF PROTOCOL MODE, THEN jump.
62AC	3A 7076		5211	LDA XMTCSR ;Get the normal Xmit CSR
62AF	A7		5212	ANA A ;Is the Xmit Rdy set?
62B0	C2 62BD		5213	JNZ CPCHO ;Jump if it is
62B3	21 7074		5214	LXI H,INTCSR ;Otherwise, point to the Intermediate CSR
62B6	22 7078		5215	SHLD XCSADR ;Store the correct Address here
62B9	23		5216	INX H ;Form the address of the correct DB
62BA	22 707A		5217	SHLD XDBADR ;Store the address here
62BD	21 7220		5218	CPCHO: LXI H,BRKFLG ;Point to this flag
62C0	7E		5219	MOV A,M ;Get it
62C1	A7		5220	ANA A ;See if set
62C2	CA 62D0		5221	JZ CPCH1 ;Jump if clear
62C5	23		5222	INX H ;Bump the pointer to the count
62C6	35		5223	DCR M ;Reduce the count
62C7	C2 62D0		5224	JNZ CPCH1 ;Jump if not 0
62CA	2B		5225	DCX H ;Point back at the flag
62CB	35		5226	DCR M ;Make it 0
62CC	3E 15		5227	MVI A,TUPREP ;Get the normal USART code
62CE	D3 47		5228	OUT TUCR ;Fix up the USART

Error	Addr	Code	Seq	Source statement
62D0	DB 83	5229	CPCH1: IN	CPATTN ;Check the CPU Attention bit
62D2	1F	5230	RAR	;by putting it in the carry
62D3	DC 6558	5231	CC	CPSERV ;Go to the service flows if set
62D6	DB 20	5232	IN	SUMREG ;Get this register
62D8	E6 40	5233	ANI	40H ;Check for 2ms done
62DA	CA 5E0C	5234	JZ	ACFAI2 ;Leave if asserted
62DD	3A 70FC	5235	LDA	PCOUNT ;Get the control P Timeout counter
62E0	A7	5236	ANA	A ;See if it is zero
62E1	CA 62ED	5237	JZ	TTRTST ;Jump if it is
62E4	3C	5238	INR	A ;Bump the count
62E5	FE FF	5239	CPI	OFFH ;See if we have timed out
62E7	CC 6625	5240	CZ	PMHALT ;Go halt the machine the hard way
62EA	32 70FC	5241	STA	PCOUNT ;Save the new count and fall through
		5242		
62ED	3A 40D7	5243	TTRTST: LDA	SPEND ;Get the ^S pending and sent flags
62F0	1F	5244	RAR	;Are we trying to put ^S in the RCV DB?
62F1	DC 6531	5245	CC	SENDS ;Go try to do it
62F4	DA 62FE	5246	JC	TTRTSB ;If the call succeeded, don't check the Q flag
62F7	3A 40D8	5247	LDA	QPEND ;Get the ^Q pending flag
62FA	1F	5248	RAR	;Is it set?
62FB	DC 6549	5249	CC	SENDQ ;Go try to send it if set
62FE	DB 06	5250	TTRTSB: IN	REMOTE ;Get the Remote/Local Switch position
6300	1F	5251	RAR	;Put it in the carry
6301	D2 64F2	5252	JNC	TTRREM ;Jump if in Remote
6304	3A 40D0	5253	LDA	SILFLG ;Get the Silo Flag
6307	1F	5254	RAR	;Anything in it?
6308	DA 6319	5255	JC	TTRTSA ;Jump if there is
630B	3A 40D9	5256	LDA	MIPFLG ;Get the Modem In Progress flag
630E	A7	5257	ANA	A ;Is it set?
630F	C2 6319	5258	JNZ	TTRTSA ;Jump if it is
6312	3A 4081	5259	LDA	SWPOS ;We must see if this flag has been set to Local
6315	A7	5260	ANA	A ;Check for 1=Local or 0=Remote.
6316	C2 6328	5261	JNZ	TTRTS0 ;Skip next part if in Local.
6319	CD 0046	5262	TTRTSA: CALL	GS_VECTOR ;Keep the modem stuff going
631C	D2 6375	5263	JNC	CPCHK2 ;Jump over next part if no characters found
631F	79	5264	MOV	A C ;Get the SR and source flags
6320	E6 C0	5265	ANI	0C0H ;Look for any remote input
6322	C2 6375	5266	JNZ	CPCHK2 ;Skip the next part if it was a remote origin
		5267		;character
6325	C3 633D	5268	JMP	TTRTS2 ;Go share the rest of the code
6328	DB 49	5269	TTRTS0: IN	TTSR ;Get the Terminal USART SR
632A	4F	5270	TTRTS1: MOV	C,A ;Save it
632B	E6 02	5271	ANI	02H ;Check RCVR Done
632D	CA 6375	5272	JZ	CPCHK2 ;Jump if Done not set.
6330	DB 48	5273	IN	TTDB ;Get the character
6332	47	5274	MOV	B,A ;save it in a temp
6333	79	5275	MOV	A,C ;Get the SR back
6334	E6 38	5276	ANI	38H ;Look for errors
6336	CA 633D	5277	JZ	TTRTS2 ;Jump if none
6339	3E 35	5278	MVI	A,35H ;Constant to be used to clear error

Error	Addr	Code	Seq	Source statement	
633B	D3 4B	5279		OUT TTCR	;Send it to the USART
633D	78	5280	TTRTS2:	MOV A,B	;Get the character back
633E	E6 7F	5281		ANI 07FH	;Mask off Parity bit
6340	FE 10	5282		CPI CNTRLP	;Is it a Control P?
6342	C2 6355	5283		JNZ TTRTS4	;Jump if not Control P
6345	DB 03	5284		IN ALLOWP	;Check for Allow Control P
6347	1F	5285		RAR	;Put it in the carry
6348	D2 6355	5286		JNC TTRTS4	;Jump if not asserted
634B	D3 AE	5287		OUT SETHLT	;Set the Halt bit
634D	3E 01	5288		MVI A,1	;Set up a timeout count
634F	32 70FC	5289		STA PCOUNT	;for control P in progress
6352	C3 6375	5290		JMP CPCHK2	;Go to the next CPU Attention check
6355	21 40D7	5291	TTRTS4:	LXI H,SPEND	;Point to the ^S Taken/Sent/Pending flags.
6358	7E	5292		MOV A,M	;Get them.
6359	E6 03	5293		ANI 3	;Save ^S (Sent OR Pending).
635B	57	5294		MOV D,A	;Put the result here for a moment.
635C	23	5295		INX H	;Point to the ^Q Pending/Sent flag byte
635D	7E	5296		MOV A,M	;Get them.
635E	E6 02	5297		ANI 2	;Save only ^Q Sent.
6360	B2	5298		ORA D	;OR it in
6361	C2 6375	5299		JNZ CPCHK2	;Don't overwrite the buffer if my ^S or ^Q are
		5300			;in it.
6364	21 7073	5301		LXI H,RCVDB	;Point to the Pseudo DB
6367	70	5302		MOV M,B	;Save the character
6368	2B	5303		DCX H	;Back up the pointer
6369	7E	5304		MOV A,M	;Get the old CSR
636A	E6 0.	5305		ANI 02H	;Check RCVR Done in the previous CSR
636C	79	5306		MOV A,C	;Get the new CSR back
636D	CA 6372	5307		JZ TTRTS5	;Jump if Done isn't set
6370	F6 10	5308		ORI 010H	;If it was, set the O.E. in the new CSR
6372	B6	5309	TTRTS5:	ORA M	;OR in the old CSR
6373	A1	5310		ANA C	;AND the old+new with the new CSR
6374	77	5311		MOV M,A	;Save the new CSR
		5312			
6375	DB 83	5313	CPCHK2:	IN CPATTN	;Get the CPU Attention bit
6377	1F	5314		RAR	;Rotate it into the Carry
6378	DC 6558	5315		CC CPSERV	;Call CPU Service if set
		5316			
637B	3A 7072	5317	RCVTTR:	LDA RCVCSR	;Get the Pseudo CSR
637E	E6 02	5318		ANI 02H	;Check the RCVR Done bit
6380	CA 639A	5319		JZ CPCHK3	;Jump if not set
6383	11 7070	5320		LXI D,ISR	;Point to the ISR
6386	1A	5321		LDAX D	;Get the ISR
6387	17	5322		RAL	;ISR bit 07 is the Terminal RCVR I.S. bit
6388	DA 639A	5323		JC CPCHK3	;If set it locks out further checking
638B	17	5324		RAL	;If not set, check the I.E. bit
638C	D2 639A	5325		JNC CPCHK3	;Jump if the Terminal RCVR I.E. is not set
638F	D3 AA	5326		OUT SETATN	;Set the interrupt flag to the CPU
6391	1A	5327		LDAX D	;Get the ISR again
6392	F6 80	5328		ORI 080H	;Set bit 07, the Terminal RCVR I.S. bit

Error	Addr	Code	Seq	Source statement	
	6394	12	5329	STAX D	;Put it back
	6395	13	5330	INX D	;Point to the IPR
	6396	1A	5331	LDAX D	;Get the IPR
	6397	F6 20	5332	ORI 020H	;Set bit 05, PRI 3
	6399	12	5333	STAX D	;Put it back
			5334		
	639A	DB 83	5335	CPCHK3: IN CPATTN	;Check the CPU Attention bit
	639C	1F	5336	RAR	;Put it in the carry
	639D	DC 6558	5337	CC CPSERV	;If set, go service the CPU
			5338		
	63A0	11 7070	5339	TTXTST: LXI D,ISR	;Point to the Interrupt Summary Register
	63A3	1A	5340	LDAX D	;Get the ISR
	63A4	E6 30	5341	ANI 030H	;Save only TTX I.S. and I.E.
	63A6	FE 10	5342	CPI 010H	;See if only I.E. is set
	63A8	C2 63BE	5343	JNZ CPCHK4	;Jump if I.E. not set or I.S. set
	63AB	2A 7078	5344	LHLD XCSADR	;Get the correct Xmit CSR Address
	63AE	7E	5345	MOV A,M	;Get the CSR contents
	63AF	A7	5346	ANA A	;Is the Xmit Rdy set?
	63B0	CA 63BE	5347	JZ CPCHK4	;Jump if not
	63B3	D3 AA	5348	OUT SETATN	;Set Console Attention
	63B5	1A	5349	LDAX D	;Get the ISR
	63B6	F6 20	5350	ORI 020H	;Set the Terminal XMIT I.S. bit
	63B8	12	5351	STAX D	;Save it
	63B9	13	5352	INX D	;Point to the Interrupt Priority Register
	63BA	1A	5353	LDAX D	;Get the IPR
	63BB	F6 10	5354	ORI 010H	;Set PRI 4 in the IPR
	63BD	12	5355	STAX D	;Save it
			5356		
	63BE	DB 83	5357	CPCHK4: IN CPATTN	;Check for CPU attention
	63C0	1F	5358	RAR	;Is it set?
	63C1	DC 6558	5359	CC CPSERV	;Go service it if it is.
			5360		
	63C4	21 7074	5361	SNDTTX: LXI H,INTCSR	;Point to this CSR
	63C7	7E	5362	MOV A,M	;Get it
	63C8	A7	5363	ANA A	;Is the Xmit Rdy set?
	63C9	C2 63E0	5364	JNZ SNDTTO	;Jump if it is (no character waiting)
	63CC	23	5365	INX H	;Bump the pointer twice
	63CD	46	5366	MOV B,M	;And get the character on the way
	63CE	23	5367	INX H	
	63CF	7E	5368	MOV A,M	;Get the normal Xmit CSR
	63D0	A7	5369	ANA A	;Is the Xmit Rdy set?
	63D1	CA 63E0	5370	JZ SNDTTO	;Jump if not
	63D4	35	5371	DCR M	;Make it zero
	63D5	22 7078	5372	SHLD XCSADR	;Set up this address
	63D8	23	5373	INX H	;Point to the normal Xmit DB
	63D9	70	5374	MOV M,B	;Store the character
	63DA	22 707A	5375	SHLD XDBADR	;Set up this address
	63DD	32 7074	5376	STA INTCSR	;Set the Xmit Rdy in the Intermediate CSR
	63E0	DB 06	5377	SNDTTO: IN REMOTE	;Get the Switch position
	63E2	1F	5378	RAR	;Put the flag in the carry

Error Addr	Code	Seq	Source statement	
63E3	DA 6494	5379	JC	SNDTL ;Go send to local if not in remote
63E6	3A 7E04	5380	LDA	TLKFLG ;Get the Talk Flag
63E9	1F	5381	RAR	;Is it set?
63EA	DA 6441	5382	JC	PMTALK ;Jump if set
63ED	21 7074	5383	SNDTT1: LXI	H,INTCSR ;Point to my Intermediate CSR
63F0	7E	5384	MOV	A,M ;Get it
63F1	A7	5385	ANA	A ;Is the Xmit Rdy set
63F2	CA 63FD	5386	JZ	SNDTT2 ;Jump if not (0 indicates presence of chr)
63F5	21 7076	5387	LXI	H,XMTCSR ;Point to the Xmit CSR
63F8	7E	5388	MOV	A,M ;Get the Pseudo SR
63F9	A7	5389	ANA	A ;Test the Xmit Rdy bit
63FA	C2 64A9	5390	JNZ	CPCHK5 ;Jump if it is set (no character waiting)
63FD	3A 40BE	5391	SNDTT2: LDA	LRMASK ;Get the Local/Remote Mask
6400	E6 0F	5392	ANI	0FH ;Save the low nibble only
6402	FE 01	5393	CPI	1 ;See if Local only set
6404	CA 649C	5394	JZ	SNDTLO ;Jump if it is
6407	FE 02	5395	CPI	2 ;See if APT enabled
6409	CA 6468	5396	JZ	SNDTA ;Jump if it is
640C	FE 05	5397	CPI	5 ;See if Remote and Local.
640E	CA 641E	5398	JZ	SNDTRL ;Jump if it is
6411	DB 03	5399	IN	SECURE ;Are we in the (Remote) Secure position.
6413	1F	5400	RAR	;This is also known as ALLOWP.
6414	DA 6475	5401	JC	SNDTR ;IF NOT, THEN jump to Send Terminal Remote.
6417	3A 7E67	5402	LDA	OVERRIDE_LCS ;ELSE, check for Override Local Copy in Secure.
641A	1F	5403	RAR	;Is it set?
641B	DA 6475	5404	JC	SNDTR ;IF SET, THEN jump.
		5405		;ELSE, fall through to send to Local and Remote.
641E	3A 7E10	5406	SNDTRL: LDA	RDCON ;Get the RD connect status
6421	1F	5407	RAR	;Are we conected?
6422	DA 642C	5408	JC	SNTRL1 ;Jump if we are
6425	3A 7E05	5409	LDA	DISCRD ;Get the Discard Output bit
6428	1F	5410	RAR	;Is it set?
6429	DA 649C	5411	JC	SNDTLO ;Ju if it is
642C	11 7E02	5412	SNTRL1: LXI	D,REMXSR ;Point to the Remote SR
642F	DB 49	5413	IN	TISR ;Get the Local SR
6431	47	5414	MOV	B,A ;Save it
6432	1A	5415	LDAX	D ;Get the Remote SR
6433	A0	5416	ANA	B ;AND them together
6434	1F	5417	RAR	;Put the LSB into the carry
6435	D2 64A9	5418	JNC	CPCHK5 ;Jump if both are not set
6438	AF	5419	XRA	A ;Make a 0
6439	12	5420	STAX	D ;Zero the Xmit Ready bit
643A	13	5421	INX	D ;Bump the pointer ahead to the Remote DB
643B	23	5422	INX	H ;Point to the DB
643C	7E	5423	MOV	A,M ;Get the character
643D	12	5424	STAX	D ;Store it
643E	C3 64A4	5425	JMP	SNDTL1 ;Go finish sending to Local
		5426		
6441	21 7E13	5427	PMTALK: LXI	H,TALKSR ;Point to the Talk SR
6444	7E	5428	MOV	A,M ;Get it

Error	Addr	Code	Seq	Source statement	
644	1F		5429	RAR	;Is the Ready bit asserted?
6446	D2 6458		5430	JNC PMTLK0	;Jump if not (go try to send the character)
6449	CD 0046		5431	CALL GS_VECTOR	;Go get the character
644C	D2 64A9		5432	JNC CPCHK5	;Jump if no character to process
644F	21 7E14		5433	LXI H,TALKDB	;Point to the Talk DB
6452	70		5434	MOV M,B	;Store the character
6453	2B		5435	DCX H	;Point to the Talk SR
6454	79		5436	MOV A,C	;Get the source flag
6455	E6 80		5437	ANI 80H	;Save only the RD one and clear Xmit Rdy
6457	77		5438	MOV M,A	;Store it in TALKSR
6458	3A 7E04		5439	PMTLK0: LDA TLKFLG	;Get the the Talk/Talk Echo flag back
645B	E6 02		5440	ANI 2	;Look for Talk Echo
645D	C2 641E		5441	JNZ SNDTRL	;Jump if Echo (send to both)
6460	7E		5442	MOV A,M	;Get the source flag
6461	17		5443	RAL	;See what the source was
6462	D2 6475		5444	JNC SNDTR	;If source was Local, send to remote
6465	C3 649C		5445	JMP SNDTLO	;Otherwise, send to local
			5446		
			5447		
6468	DB 41		5448	SNDTA: IN TRSR	;Look for APT Xmit Ready
646A	1F		5449	RAR	;Is it set?
646B	D2 64A9		5450	JNC CPCHK5	;Jump if not
646E	23		5451	INX H	;Point to the DB
646F	7E		5452	MOV A,M	;Get the character
6470	D3 40		5453	OUT TRDB	;Send it out
6472	C3 64A6		5454	JMP SNDTL2	;Go set the Xmit Ready in the Pseudo SR
			5455		
6475	3A 7E10		5456	SNDTR: LDA RDCON	;Get the RD connect status
6478	1F		5457	RAR	;Are we conected?
6479	DA 6483		5458	JC SNDTR1	;Jump if we are
647C	3A 7E05		5459	LDA DISCRD	;Get the Discard Output bit
647F	1F		5460	RAR	;Is it set?
6480	DA 64A7		5461	JC SNDTL3	;Jump if it is
6483	11 7E02		5462	SNDTR1: LXI D,REMXSR	;Point to the Remote Xmit SR
6486	1A		5463	LDAX D	;Get it
6487	1F		5464	RAR	;Is Xmit Ready set?
6488	D2 64A9		5465	JNC CPCHK5	;Jump if not
648B	AF		5466	XRA A	;Make a zero
648C	12		5467	STAX D	;Zero the Xmit Ready
648D	13		5468	INX D	;Point the the DB
648E	23		5469	INX H	;Point to the DB
648F	7E		5470	MOV A,M	;Get the character
6490	12		5471	STAX D	;Store the character
6491	C3 64A6		5472	JMP SNDTL2	;Go set the Xmit Ready in the Pseudo SR
			5473		
			5474		
6494	21 7076		5475	SNDTL: LXI H,XMTCSR	;Point to the Xmit CSR
6497	7E		5476	MOV A,M	;Get the CSR
6498	1F		5477	RAR	;Is the Xmit Ready bit set?
6499	DA 64A9		5478	JC CPCHK5	;Jump if it is (means no character waiting)

Error	Addr	Code	Seq	Source statement	
649C	DB 49	5479	SNDTLO: IN	TTSR	;Get the Local SR
649E	1F	5480	RAR		;Check for Xmit Ready
649F	D2 64A9	5481	JNC	CPCHK5	;Jump if not ready
64A2	23	5482	INX	H	;Point to the character
64A3	7E	5483	MOV	A,M	;Get the character
64A4	D3 48	5484	SNDTL1: OUT	TTDB	;Send it to the local DB
64A6	2B	5485	SNDTL2: DCX	H	;Back the pointer up
64A7	36 01	5486	SNDTL3: MVI	M,1	;Set the Xmit Ready
		5487			
64A9	DB 83	5488	CPCHK5: IN	CPATTN	;Check CPU attention
64AB	1F	5489	RAR		;Rotate it into the carry
64AC	DC 6558	5490	CC	CPSERV	;Call if set
		5491			
64AF	11 7070	5492	TUXTST: LXI	D,ISR	;Point to the ISR
64B2	1A	5493	LDAX	D	;Get the ISR
64B3	E6 0C	5494	ANI	0CH	;Save only the TUX I.E. and I.S. bits
64B5	FE 08	5495	CPI	08H	;See if I.E. set and I.S. clear
64B7	C2 64CB	5496	JNZ	CPCHK6	;Jump if not
64BA	DB 45	5497	IN	TUSR	;Get the TU-58 USART SR
64BC	1F	5498	RAR		;Check the Ready bit
64BD	D2 64CB	5499	JNC	CPCHK6	;Jump if not set
64C0	D3 AA	5500	OUT	SETATN	;Set the Console Attention flag
64C2	1A	5501	LDAX	D	;Get the ISR
64C3	F6 04	5502	ORI	04H	;Set bit 02, the TU-58 XMIT ISR bit
64C5	12	5503	STAX	D	;Put it away
64C6	13	5504	INX	D	;Point to the IPR
64C7	1A	5505	LDAX	D	;Get the IPR
64C8	F6 40	5506	ORI	040H	;Set the PRI 2 Flag
64CA	12	5507	STAX	D	;Put it away
		5508			
64CB	DB 83	5509	CPCHK6: IN	CPATTN	;Check CPU attention
64CD	1F	5510	RAR		;Rotate it into the carry
64CE	DC 6558	5511	CC	CPSERV	;Call if set
		5512			
64D1	DB 45	5513	TURTST: IN	TUSR	;Get the TU-58 SR
64D3	E6 02	5514	ANI	02H	;Check the Done bit
64D5	CA 629D	5515	JZ	CPCHK1	;Jump if not Done
64D8	11 7070	5516	LXI	D,ISR	;Point to the ISR
64DB	1A	5517	LDAX	D	;Get the ISR
64DC	1F	5518	RAR		;Check the TUR Interrupt Status
64DD	DA 629D	5519	JC	CPCHK1	;Jump if already set
64E0	1F	5520	RAR		;Check the I.E.
64E1	D2 629D	5521	JNC	CPCHK1	;Jump if not set
64E4	D3 AA	5522	OUT	SETATN	;Otherwise, set Console Attention
64E6	1A	5523	LDAX	D	;Get the ISR
64E7	F6 01	5524	ORI	01H	;Set the TUR ISR BIT
64E9	12	5525	STAX	D	;Put the ISR back
64EA	13	5526	INX	D	;Point to the IPR
64EB	1A	5527	LDAX	D	;Get the IPR
64EC	F6 80	5528	ORI	080H	;Set Priority 1

Error Addr	Code	Seq	Source statement	
64EE	12	5529	STAX D	;Put the IPR back
64EF	C3 629D	5530	JMP CPCHK1	;We're done
		5531		
		5532		
64F2	3A 7E04	5533	TTRREM: LDA TLKFLG	;Get the Talk Flag
64F5	A7	5534	ANA A	;Is it set?
64F6	C2 652D	5535	JNZ CHECK_MODEM	;Skip the next part if it is
64F9	3A 7E05	5536	LDA DISCRD	;Get the Discard Remote Output bit
64FC	21 7E10	5537	LXI H,RDCON	;Point to the RD Connected flag
64FF	B6	5538	ORA M	;OR it in
6500	EE 01	5539	XRI 1	;Flip the result
6502	23	5540	INX H	;Point to flag that indicates past connection
6503	A6	5541	ANA M	;AND it in
6504	47	5542	MOV B,A	;Store it in B for a moment.
6505	21 40D7	5543	LXI H,SPEND	;See what the ^S status is.
6508	7E	5544	MOV A,M	;Get the ^S Taken/Sent/Pending flags.
6509	E6 03	5545	ANI 3	;Save only Sent and Pending.
650B	4F	5546	MOV C,A	;Put it in C.
650C	23	5547	INX H	;Point at the ^Q Sent/Pending flags.
650D	E6 02	5548	ANI 2	;Save only the Sent (^Q in the RCVDB) flag.
650F	B0	5549	ORA B	;OR everything together.
6510	B1	5550	ORA C	;
6511	C2 652D	5551	JNZ CHECK_MODEM	;Don't touch Silo if (-RDCON*-DISCRD*RDCON1)
		5552		;OR (^S (Sent OR Pending)) OR ^Q Sent.
6514	21 40F4	5553	TTRRE1: LXI H,SLOW_CLK_FLG	;Point to this flag.
6517	DB 06	5554	IN SLWCLK	;Get the Slow Clock
6519	E6 80	5555	ANI 80H	;Save only the MSB.
651B	BE	5556	CMP M	;Are the Flag and the Clock the same?
651C	CA 652D	5557	JZ CHECK_MODEM	;IF EQUAL, THEN jump (5ms hasn't elapsed).
651F	77	5558	MOV M,A	;ELSE, create the new Slow Clock Flag.
6520	CD 0046	5559	CALL GS_VECTOR	;Now go get the character
6523	D2 6375	5560	JNC CPCHK2	;Jump if none found
6526	79	5561	MOV A,C	;Get the SR
6527	E6 3F	5562	ANI 3FH	;Clear bits 7 and 6
6529	4F	5563	MOV C,A	;Restore it
652A	C3 633D	5564	JMP TTRTS2	;Go share processing code
		5565		
652D		5566	CHECK_MODEM:	
652D	E7	5567	RST 4	;Keep the Modem servicing frequency up
652E	C3 6375	5568	JMP CPCHK2	;Go back to the normal flow.
		5569		
		5570		
6531	0E 13	5571	SENDS: MVI C,CNTRLS	;Set up a ^S in C
6533	CD 653E	5572	CALL SNDCTL	;Go try to send the control S
6536	C0	5573	RNZ	;Leave if the ^S was not sent
6537	21 40D7	5574	LXI H,SPEND	;Point to the ^S flags
653A	36 02	5575	MVI M,2	;Set bit 1 and clear bit 0
653C	37	5576	STC	;Set the carry
653D	C9	5577	RET	;Leave
		5578		

Error	Addr	Code	Seq	Source statement	
653E	21 7072		5579	SNDCTL: LXI	H,RCVCSR ;Point to the Pseudo Rcv CSR
6541	7E		5580	MOV	A,M ;Get it
6542	A0		5581	ANA	B ;Is the Done bit set?
6543	C0		5582	RNZ	
6544	36 02		5583	MVI	M,2 ;Leave if it is
6546	23		5584	INX	H ;Set the Rcvr Done bit
6547	71		5585	MOV	M,C ;Point to the DB
6548	C9		5586	RET	
			5587		;Put the correct control character there
			5588		;Leave with the Z bit set
6549	0E 11		5588	SENDQ: MVI	C,CNTRLQ ;Set up a ^Q in C
654B	CD 653E		5589	CALL	SNDCTL ;Go share the next part
654E	C0		5590	RNZ	
654F	21 40D7		5591	LXI	H,SPEND ;Leave if we couldn't post the ^Q
6552	36 00		5592	MVI	M,0 ;Point to the ^S flags
6554	23		5593	INX	H ;Zero them out
6555	36 02		5594	MVI	M,2 ;Point to the ^Q flags
6557	C9		5595	RET	
			5596		;Set bit 1 and clear bit 0
			5597		;Leave
			5598		
6558	DB 80		5599	CPSEV: IN	READ ;Get the offset from the CPU
655A	6F		5600	MOV	L,A ;Put it in register L
655B	D3 A8		5601	OUT	SETACK ;Set Console Acknowledge
655D	1E 00		5602	MVI	E,0 ;Clear register E for timeout counting
655F	63		5603	MOV	H,E ;Clear register H for the DAD at CPSEV2
6560	DB 83		5604	CPSEV1: IN	CPATTN ;Wait for CPU Attention to go away
6562	1F		5605	RAR	
6563	D2 656D		5606	JNC	CPSEV2 ;Rotate it into the carry
6566	1C		5607	INR	E ;Exit loop if CPU Attention goes away
6567	CC 5D5E		5608	CZ	COMERR ;Bump the Timeout count
656A	C3 6560		5609	JMP	CPSEV1 ;If the count reaches 0 it is an error
656D	11 70A2		5610	CPSEV2: LXI	D,TABLE ;Loop on CPU Attention
6570	19		5611	DAD	D ;Get the address of the Table
6571	11 7070		5612	LXI	D,ISR ;Add it to the offset
6574	E9		5613	PCHL	
			5614		;Point to the ISR
			5615		;Decode the command
6575	13		5616	INTACK: INX	D ;Point to the IPR
6576	1A		5617	LDAX	D ;Get the IPR
6577	D3 CC		5618	OUT	WRITE ;Put the data in the WRITE Register
6579	D3 AB		5619	OUT	CLRATN ;Console Attention must be cleared first
			5620		;in case this Interrupt Acknowledge
			5621		;has only one interrupt pending
657B	D3 A9		5622	OUT	CLRACK ;Clear Console Ack second
657D	17		5623	RAL	
657E	DA 66A5		5624	JC	PRI1 ;Check PRI 1
6581	17		5625	RAL	
6582	DA 66AA		5626	JC	PRI2 ;Go take care of Priority 1 device
6585	17		5627	RAL	
6586	DA 66AF		5628	JC	PRI3 ;Check PRI 2
					;Go take care of Priority 2 device
					;Check PRI 3
					;Go take care of Priority 3 device

Error	Addr	Code	Seq	Source statement	
	6589	C3 66B4	5629	JMP	PRI4
			5630		
			5631		
	658C	2A 7078	5632	TTXCSR: LHLD	XCSADR
	658F	7E	5633	MOV	A,M
	6590	D3 CC	5634	LAST: OUT	WRITE
	6592	D3 A9	5635	OUT	CLRACK
	6594	C9	5636	RET	
			5637		
			5638		
	6595	3A 7072	5639	TTRCSR: LDA	RCVCSR
			5640		
	6598	C3 6590	5641	JMP	LAST
			5642		
	659B	DB 45	5643	TURCSR: IN	TUSR
	659D	C3 6590	5644	JMP	LAST
			5645		
	65A0	DB 80	5646	TTXDB: IN	READ
	65A2	D3 A9	5647	OUT	CLRACK
	65A4	2A 707A	5648	LHLD	XDBADR
	65A7	77	5649	MOV	M,A
	65A8	2B	5650	DCX	H
	65A9	36 00	5651	MVI	M,0
	65AB	1A	5652	LDAX	D
	65AC	E6 DF	5653	ANI	0DFH
	65AE	12	5654	STAX	D
	65AF	C9	5655	RET	
			5656		
	65B0	DB 80	5657	TUXDB: IN	READ
	65B2	D3 44	5658	OUT	TUDB
	65B4	D3 A9	5659	OUT	CLRACK
	65B6	1A	5660	LDAX	D
	65B7	E6 FB	5661	ANI	0FBH
	65B9	12	5662	STAX	D
	65BA	C9	5663	RET	
			5664		
	65BB	D3 A9	5665	TTXIE: OUT	CLRACK
	65BD	1A	5666	LDAX	D
	65BE	F6 10	5667	ORI	010H
	65C0	12	5668	STAX	D
	65C1	C9	5669	RET	
			5670		
	65C2	D3 A9	5671	TTRIE: OUT	CLRACK
	65C4	1A	5672	LDAX	D
	65C5	F6 40	5673	ORI	040H
	65C7	12	5674	STAX	D
	65C8	C9	5675	RET	
			5676		
	65C9	DB 80	5677	TUWCSR: IN	READ
	65CB	D3 A9	5678	OUT	CLRACK

;Go clear PRI 4

;Get the address of the correct CSR

;Get the correct CSR

;Write Register A to the CPU

;Clear Console Ack to tell the CPU it's there

;Leave

;MFPR's for the Terminal RCVR CSR actually

;access the Pseudo CSR

;This code will finish up the transfer

;MFPR's for the TU-53 CSR get the USART SR

;Go finish the transfer

;Read the character to be sent to the Terminal

;Signal the CPU that we got it

;Get the address of the correct Xmit DB

;Store the character

;Point to the correct Xmit CSR

;Clear the Xmit Ready

;Get the ISR

;Clear bit 05, the Terminal XMIT I.S. bit

;Put the ISR back

;We're done

;Get the character to be sent to the TU-58

;Write it

;Tell the CPU we got the character

;Get the ISR

;Clear bit 02, the TU-58 XMIT I.S. bit

;Put it in the ISR

;We're done

;Tell the CPU that we got the command

;Get the ISR

;Set the Terminal Xmit I.E. in the ISR

;Put the ISR back

;leave

;Tell the CPU that we got it

;Get the ISR

;Set the Terminal Rcvr I.E.

;Put the ISR back

;Leave

;Get the next byte

;Clear Console Acknowledge

Error	Addr	Code	Seq	Source statement	
65CD	E6 41		5679	ANI 41H	;Save only the I.E. and Break bits
65CF	1F		5680	RAR	;Put the LSB (Break bit) in the carry
65D0	47		5681	MOV B,A	;Save the rotated version, I.E. in bit 5
65D1	D2 65E0		5682	JNC TUWCS1	;Jump if no Break
65D4	21 7220		5683	LXI H,BRKFLG	;Point to the flag
65D7	36 01		5684	MVI M,1	;Set it
65D9	23		5685	INX H	;Point to the count
65DA	36 03		5686	MVI M,3	;Set up a count
65DC	3E 0D		5687	MVI A,BREAK	;Get the code for a break
65DE	D3 47		5688	OUT TUCR	;Send it to the USART
65E0	AF		5689	TUWCS1: XRA A	;Clear the carry
65E1	78		5690	MOV A,B	;Get the rotated version back
65E2	1F		5691	RAR	;Rotate it into bit 4
65E3	1F		5692	RAR	;and now into bit 3
65E4	47		5693	MOV B,A	;Save this version
65E5	0E 08		5694	MVI C,8	;Set up register C for comparing and such
65E7	1A		5695	LDAX D	;Get the ISR to check the old I.E.
65E8	A1		5696	ANA C	;Save the old TUXIE only
65E9	B8		5697	CMP B	;See if they are the same
65EA	C8		5698	RZ	;Leave if they are
65EB	78		5699	MOV A,B	;Put the new one in A
65EC	B9		5700	CMP C	;See if the new I.E. is set or clear
65ED	C2 6611		5701	JNZ TUXID	;Go clear it
65F0	EB		5702	XCHG	;Put the pointer in HL instead of DE
65F1	B6		5703	ORA M	;Set the I.E. in the ISR
65F2	77		5704	MOV M,A	;Save the new version
65F3	C9		5705	RET	;Leave
			5706		
65F4	D3 A9		5707	TURIE: OUT CLRACK	;Clear Console Acknowledge
65F6	1A		5708	LDAX D	;Get the ISR
65F7	F6 02		5709	ORI 02H	;Set the TU-58 Rcv I.E.
65F9	12		5710	STAX D	;Put the ISR back
65FA	C9		5711	RET	;Leave
			5712		
65FB	D3 AB		5713	TTXID: OUT CLRATN	;Clear Console attention
65FD	D3 A9		5714	OUT CLRACK	;Clear Console Ack
65FF	1A		5715	LDAX D	;Get the ISR
6600	E6 CF		5716	ANI 0CFH	;Clear bits <05:04>
6602	12		5717	STAX D	;Put the ISR back
6603	C3 66B4		5718	JMP PRI4	;Go clear Priority 4
			5719		
			5720		
6606	D3 AB		5721	TTRID: OUT CLRATN	;Clear Console Attention
6608	D3 A9		5722	OUT CLRACK	;Clear Console Ack
660A	1A		5723	LDAX D	;Get the ISR
660B	E6 3F		5724	ANI 03FH	;Clear bits <07:06>
660D	12		5725	STAX D	;Put the ISR back
660E	C3 66AF		5726	JMP PRI3	;Go clear Priority 3
			5727		
			5728		

Error Addr	Code	Seq	Source statement
6611	D3 AB	5729	TUXID: OUT CLRATN ;Clear Console Attention
6613	1A	5730	LDAX D ;Get the ISR
6614	E6 F3	5731	ANI 0F3H ;Clear bits <03:02>
6616	12	5732	STAX D ;Put the ISR back
6617	C3 66AA	5733	JMP PRI2 ;Go clear Priority 2
		5734	
661A	D3 AB	5735	TURID: OUT CLRATN ;Clear the Console Attention bit
661C	D3 A9	5736	OUT CLRACK ;Clear Console Ack
661E	1A	5737	LDAX D ;Get the ISR
661F	E6 FC	5738	ANI 0FCH ;Clear bits <01:00>
6621	12	5739	STAX D ;Put the ISR back
6622	C3 66A5	5740	JMP PRI1 ;Go clear Priority 1
		5741	
		5742	
6625	D3 A9	5743	PMHALT: OUT CLRACK ;Clear Console Ack
6627	D3 AB	5744	OUT CLRATN ;Turn off Console Attention
6629	D3 AF	5745	OUT CLRHLT ;Clear the Halt bit
662B	D3 3C	5746	OUT CLRLIT ;Turn off the Run light
662D	AF	5747	XRA A ;Make a 0
662E	32 40D3	5748	STA RUNFLG ;Turn off the Run Flag
6631	E1	5749	POP H ;Fix up the stack
6632	3A 40B5	5750	LDA PFFLAG ;Get the Power Fail flag
6635	A7	5751	ANA A ;Is it set?
6636	C0	5752	RNZ ;Leave if it is
6637	3A 7076	5753	LDA XMTCSR ;Get the pseudo XMT CSR
663A	A7	5754	ANA A ;Is Xmit Rdy set?
663B	C2 6647	5755	JNZ PMHALO ;Jump if it is (means no character in DB)
663E	3A 7077	5756	LDA XMTDB ;Get the character
6641	DF	5757	RST 3 ;Go print it
6642	3E 01	5758	MVI A,1 ;Make a 1
6644	32 7076	5759	STA XMTCSR ;Set the Xmit Rdy
6647	3A 7074	5760	PMHALO: LDA INTCSR ;Get the INT CSR
664A	A7	5761	ANA A ;Is Xmit Rdy set?
664B	C0	5762	RNZ ;Leave if not (means no character in DB)
664C	3A 7075	5763	LDA INTDB ;Get the character
664F	DF	5764	RST 3 ;Go print it
6650	3E 01	5765	MVI A,1 ;Make a 1
6652	32 7074	5766	STA INTCSR ;Set the Xmit Rdy
6655	C9	5767	RET ;Leave
		5768	
		5769	
6656	1A	5770	TTRDB: LDAX D ;Get the ISR
6657	E6 7F	5771	ANI 07FH ;Clear Terminal Rcvr I.S.
6659	12	5772	STAX D ;Store it
665A	21 7073	5773	LXI H,RCVDB ;Point to the Pseudo DB
665D	7E	5774	MOV A,M ;Get the Pseudo DB
665E	D3 CC	5775	OUT WRITE ;Send the character to the CPU
6660	CD 6971	5776	CALL WAITH ;Go wait for CPATTN to go high
6663	2B	5777	DCX H ;Back up the pointer
6664	7E	5778	MOV A,M ;Send the CSR next. The CSR is sent because

Error Addr	Code	Seq	Source statement	
		5779		;it contains the Error bits
6665	D3 CC	5780	OUT WRITE	;Send it
6667	CD 6980	5781	CALL WAITL	;Go wait for CPU Attention to go away
666A	D3 A9	5782	OUT CLRACK	;Clear Console Ack
666C	AF	5783	XRA A	;Generate 0
666D	77	5784	MOV M,A	;This clears Done and the Errors
666E	21 40D7	5785	LXI H,SPEND	;Point to the ^S Pending/Sent flags
6671	7E	5786	MOV A,M	;Get them
6672	E6 02	5787	ANI 2	;See if this was a Console generated ^S we sent
6674	CA 6679	5788	JZ TTRDB1	;Jump if not
6677	36 04	5789	MVI M,4	;Set bit 2 to indicate the ^S was taken
6679	21 40DB	5790	TTRDB1: LXI H,QPEND	;Point to the ^Q Pending/Sent flags
667C	7E	5791	MOV A,M	;Get them
667D	FE 02	5792	CPI 2	;See if the ^Q Sent flag is set
667F	C0	5793	RNZ	;Leave if not
6680	36 00	5794	MVI M,0	;Clear it if it was
6682	C9	5795	RET	;Leave
		5796		
		5797		
6683	1A	5798	TURDB: LDAX D	;Get the ISR
6684	E6 FE	5799	ANI 0FEH	;Clear the TU-58 Rcvr I.S.
6686	12	5800	STAX D	;Put the ISR back
6687	DB 44	5801	IN TUDB	;Get the character
6689	6F	5802	MOV L,A	;Save it
668A	DB 45	5803	I TUSR	;Get the SR
668C	67	5804	MOV H,A	;Save it
668D	E6 38	5805	ANI 38H	;Check for errors
668F	CA 6696	5806	JZ TURDB0	;Jump over next part if none
6692	3E 15	5807	MVI A,015H	;Used to clear possible ERRORS
6694	D3 47	5808	OUT TUCR	;Clear them
6696	7D	5809	TURDB0: MOV A,L	;Get the character
6697	D3 CC	5810	OUT WRITE	;Send it
6699	CD 6971	5811	CALL WAITH	;Go wait for CPATTN to go nigh
669C	7C	5812	MOV A,H	;Get the error bits
669D	D3 CC	5813	OUT WRITE	;Send them
669F	CD 6980	5814	CALL WAITL	;Go wait for CPATTN to go away
66A2	D3 A9	5815	OUT CLRACK	;Clear Console Ack
66A4	C9	5816	RET	;Leave
		5817		
66A5	3E 7F	5818	PR11: MVI A,07FH	;Clear Priority 1
66A7	C3 66B6	5819	JMP IPRCHK	;Check the IPR
		5820		
66AA	3E BF	5821	PR12: MVI A,0BFH	;Clear Priority 2
66AC	C3 66B6	5822	JMP IPRCHK	;Go check the remaining priorities
		5823		
66AF	3E DF	5824	PR13: MVI A,0DFH	;Clear Priority 3
66B1	C3 66B6	5825	JMP IPRCHK	;Go do it and check remaining priorities
		5826		
66B4	3E EF	5827	PR14: MVI A,0EFH	;Set up to clear PRI 4
		5828		

Error	Addr	Code	Seq	Source statement	
66B6	21	7071	5829	IPRCHK: LXI	H,IPR
66B9	A6		5830	ANA	M
66BA	77		5831	MOV	M,A
66BB	E6	F0	5832	ANI	0F0H
66BD	C8		5833	RZ	
66BE	3A	7223	5834	LDA	IFLAG
66C1	A7		5835	ANA	A
66C2	C0		5836	RNZ	
66C3	D3	AA	5837	OUT	SETATN
66C5	C9		5838	RET	
			5839		
66C6	DB	80	5840	ITWCSR: IN	READ
66C8	D3	A9	5841	OUT	CLRACK
66CA	5F		5842	MOV	E,A
66CB	E6	07	5843	ANI	07H
66CD	57		5844	MOV	D,A
66CE	21	707C	5845	LXI	H,ICCS
66D1	7E		5846	MOV	A,M
66D2	F6	87	5847	ORI	087H
66D4	A3		5848	ANA	E
66D5	5E		5849	MOV	E,M
66D6	77		5850	MOV	M,A
66D7	17		5851	RAL	
66D8	D2	66E6	5852	JNC	ITWCS0
66DB	7E		5853	MOV	A,M
66DC	E6	60	5854	ANI	060H
66DE	CA	66E6	5855	JZ	ITWCS0
66E1	D3	A4	5856	OUT	SETTIM
66E3	C3	66E8	5857	JMP	ITWCS1
66E6	D3	A5	5858	ITWCS0: OUT	CLRTIM
66E8	7B		5859	ITWCS1: MOV	A,E
66E9	E6	01	5860	ANI	01H
66EB	32	707D	5861	STA	OLDRUN
66EE	BA		5862	CMP	D
			5863		
66EF	C8		5864	RZ	
66F0	7A		5865	MOV	A,D
66F1	1F		5866	RAR	
66F2	DA	679E	5867	JC	ITRUN
66F5	D3	2C	5868	OUT	CLRTMR
66F7	20		5869	RIM	
66F8	F6	09	5870	ORI	9H
66FA	30		5871	SHL	
66FB	7A		5872	MOV	A,D
66FC	E6	02	5873	ANI	02
66FE	CA	671F	5874	JZ	ITWCS2
6701	3A	7083	5875	LDA	NICFLG
6704	A7		5876	ANA	A
6705	C4	6823	5877	CNZ	XFRSET
6708	AF		5878	XRA	A

```

;Point to the IPR
;AND in the IPR
;Save the new IPR
;Are any PRI bits set?
;If none set, return with Console Attention off
;Get the I space operation flag
;Is it set?
;Leave if it is
;Else, set Console Attention
;and leave

;Get the new ICCS value
;Clear Console Ack
;Save the new ICCS in a tmp
;Make a copy with I.E., INT and ERR clear
;Save it in another tmp
;Point to the ICCS
;Get the ICCS
;Set everything except INT and ERR
;Generate the new CSR
;Save the old ICCS for a bit
;Install the new one
;Check for I.E.
;If not set, go clear the INTERRUPT
;Get the ICCS back
;See if either INT or ERR is set
;If neither is set, go clear Interrupt
;May or may not already be set
;Skip over CLRINT
;Clear the timer Interrupt
;Get the old ICCS
;Save only the Run bit, clear SGL and XFR
;Save the old RUN bit
;See if these bits in the new ICCS and the old
;are the same
;Leave if they are
;Something is different, so get the new ICCS
;Check the run bit
;Go start the timer, forget about SGL
;Stop the counters
;Get the interrupt masks
;Set MSE and MS.5
;Write the masks out
;Get the ICCS
;Check for XFR
;Skip if no XFR
;Is there a new count
;See if the NIC flag is set
;Go do the load
;Make a 0

```


Error	Addr	Code	Seq	Source statement	
6709	32 7099	5879	STA	IMPXFR	;Clear the Implied XFR flag
670C	3C	5880	INR	A	;Make a 1
670D	32 7098	5881	STA	BLKFLG	;Set the flag to indicate the 100Hz is blocked
6710	D3 3B	5882	OUT	SETBLK	;Block the 100Hz clock
6712	3E 63	5883	MVI	A,LARM12	;Don't do this as part of the XFRSET subroutine
6714	D3 1D	5884	OUT	ITCSR	;because XFRSET is also called from the timer
		5885			;handlers if new M1CR must be loaded on the fly
6716	AF	5886	XRA	A	;Make a 0
6717	32 7098	5887	STA	BLKFLG	;Zero the 100Hz clock blocked flag
671A	D3 3A	5888	OUT	CLRBLK	;Unblock the 100Hz clock
671C	CD 69D7	5889	CALL	TMRPR1	;Go reset any interrupts
671F	3A 707C	5890	LDA	ITWCS2: ICCS	;Get the ICCS
6722	E6 04	5891	ANI	04	;Is SGL set
6724	C8	5892	RZ		;Leave if not
6725	32 707E	5893	STA	ITSGL	;Set the flag that says an interval timer single
		5894			;step has occurred
6728	3A 7082	5895	LDA	LOADED	;Get the flag that indicates what is in the
		5896			;9513 Alarm registers
672B	A7	5897	ANA	A	;If it is zero we don't need to do anything
672C	C4 684C	5898	CNZ	NEWCNT	;Go reload the Alarm registers if LOADED<>0
672F	3E 01	5899	MVI	A,1	;Make a 1
6731	32 7098	5900	STA	BLKFLG	;Set the flag to indicate the 100Hz is blocked
6734	D3 3B	5901	OUT	SETBLK	;Block the 100Hz clock
6736	3E A3	5902	MVI	A,SAVE12	;Save the current value
6738	D3 1D	5903	OUT	ITCSR	;Do it
673A	3E 19	5904	MVI	A,HOLD1	;Point to the Counter 1 Hold register
673C	D3 1D	5905	OUT	ITCSR	;Do it
673E	DB 1C	5906	IN	ITDB	;Get the low byte of Hold
6740	3C	5907	INP	A	;Check for value=0FFH
6741	C2 6747	5908	JNZ	ITWCS3	;If not zero we don't need to look at the next
6744	DB 1C	5909	IN	ITDB	;Get the high byte
6746	3C	5910	INR	A	;Bump the high byte
6747	3E F1	5911	MVI	A,STEP1	;Step the low counter
6749	D3 1D	5912	OUT	ITCSR	;Do it
674B	C2 6752	5913	JNZ	ITWCS4	;Leave if not zero
674E	3E F2	5914	MVI	A,STEP2	;If zero, bump the high count
6750	D3 1D	5915	OUT	ITCSR	;Do it
6752	CD 69D7	5916	CALL	ITWCS4: TMRPR1	;Go try to reset the overflow
6755	AF	5917	XRA	A	;Make a 0
6756	32 7098	5918	STA	BLKFLG	;Clear the 100Hz blocked flag
6759	D3 3A	5919	OUT	CLRBLK	;Clear the 100Hz block
675B	DB 20	5920	IN	SUMREG	;See if we caused an overflow
675D	17	5921	RAL		
675E	D8	5922	RC		;Leave if not
675F	3A 707C	5923	LDA	ITWCS5: ICCS	;Get the ICCS
6762	17	5924	RAL		;Check the I.E.
6763	D2 6768	5925	JNC	ITWCS5	;Jump if no I.E.
6766	D3 A4	5926	CJT	SETTIM	;Set the Timer interrupt to the CPU
6768	3E 01	5927	MVI	A,1	;Make a 1
676A	32 7098	5928	STA	BLKFLG	;Set the flag to indicate the 100Hz is blocked

Error	Addr	Code	Seq	Source statement
676D	D3 3B	5929	OUT	SETBLK ;Block the 100Hz clock
676F	3E 63	5930	MVI	A,LARM12 ;Prepare to Load and Arm counters 1 and 2
6771	D3 1D	5931	OUT	ITCSR ;Do it
6773	AF	5932	XRA	A ;Make a 0
6774	32 7098	5933	STA	BLKFLG ;Zero the 100Hz clock blocked flag
6777	D3 3A	5934	OUT	CLRBLK ;Unblock the 100Hz clock
6779	21 7083	5935	LXI	H,NICFLG ;Point to this flag
677C	7E	5936	MOV	A,M ;Get it
677D	A7	5937	ANA	A ;Is the NIC flag set?
677E	CA 6789	5938	JZ	ITWCS6 ;Jump if not
6781	32 7099	5939	STA	IMPXFR ;Set the Implied XFR occurred flag
6784	C5	5940	PUSH	B ;Save BC
6785	CD 6823	5941	CALL	XFRSET ;Go figure out what to load and load it
6788	C1	5942	POP	B ;and BC
6789	21 707C	5943	LXI	H,ICCS ;Point to the ICCS
678C	7E	5944	MOV	A,M ;Get the ICCS
678D	E6 40	5945	ANI	040H ;Check for INT already set
678F	CA 6793	5946	JZ	ITWCS7 ;Jump if not already set
6792	1F	5947	RAR	ITWCS7: ;This will set bit 05 quickly
6793	F6 40	5948	ORI	040H ;Set the INT bit
6795	B6	5949	ORA	M ;OR in the ICCS
6796	77	5950	MOV	M,A ;Store the new ICCS
6797	3E 40	5951	MVI	A,040H ;Set SOE and clear SOD
6799	30	5952	SIM	ITWCS7: ;Drive SOD low, then high to reset Timer Int
679A	3E C0	5953	MVI	A,0C0H ;
679C	30	5954	SIM	ITWCS7: ;
679D	C9	5955	RET	ITWCS7: ;Leave
679E	1F	5956		
679F	DA 67FF	5957		
67A2	3A 707D	5958	RAR	ITRUN: ;Check for XFR
67A5	A7	5959	JC	ITRUN1 ;Jump if XFR set
67A6	C0	5960	LDA	OLDRUN ;Get the Old Run flag
67A7	CD 684C	5961	ANA	A ;Is it set?
67AA	21 707E	5962	RNZ	ITRUN: ;Leave if the timer is already running
67AD	7E	5963	CALL	NEWCNT ;Go load the correct Alarm value
67AE	A7	5964	LXI	H,ITSGL ;Find out if any single stepping has occurred
67AF	36 00	5965	MOV	A,M ;Get the flag
67B1	CA 67F4	5966	ANA	A ;Test it
		5967	MVI	M,0 ;Zero it (just in case)
		5968	JZ	ITRUN0 ;If zero we don't need to do anymore
		5969		
		5970		;Otherwise, we must find out if the value in the counters is greater than the
		5971		;minus 3 or minimum count version we have just loaded via NEWCNT. If it is,
		5972		;we must pretend that an overflow has occurred, set interrupts if necessary,
		5973		;and start the counter over from zero by doing a Load and Arm on counters 1
		5974		;and 2.
		5975		
67B4	3E 01	5976	MVI	A,1 ;Make a 1
67B6	32 7098	5977	STA	BLKFLG ;Set the flag to indicate the 100Hz is blocked
67B9	D3 3B	5978	OUT	SETBLK ;Block the 100Hz clock

Error	Addr	Code	Seq	Source statement	
	67BB	3E A3	5979	MVI A,SAVE12	;Save the present contents of counters 1 and 2
	67BD	D3 1D	5980	OUT ITCSR	;
	67BF	3E 19	5981	MVI A,HOLD1	;Point to the internal pointer to Hold Registers
	67C1	D3 1D	5982	OUT ITCSR	;1 and 2
	67C3	AF	5983	XRA A	;Make a 0
	67C4	32 7098	5984	STA BLKFLG	;Zero the 100Hz clock blocked flag
	67C7	D3 3A	5985	OUT CLRBLK	;Unblock the 100Hz clock
	67C9	11 709E	5986	LXI D,ITTEMP	;Point to this temp
	67CC	DB 1C	5987	IN ITDB	;Get the low byte of the count
	67CE	12	5988	STAX D	;Store it
	67CF	23	5989	INX H	;Bump the pointer
	67D0	DB 1C	5990	IN ITDB	;Get the next byte of the count
	67D2	12	5991	STAX D	;Store it
	67D3	23	5992	INX H	;Bump the pointer
	67D4	DB 1C	5993	IN ITDB	;Get the next byte of the count
	67D6	12	5994	STAX D	;Store it
	67D7	DB 1C	5995	IN ITDB	;Get the next byte of the count
	67D9	21 7097	5996	LXI H,ALRMM3+3	;Point to the high byte of what is in Alarms
	67DC	BE	5997	CMP M	;Check for Counter<Alarm
	67DD	DA 67F4	5998	JC ITRUN0	;Jump if Counter<Alarm
	67E0	1A	5999	LDAX D	;Get the next counter byte
	67E1	2B	6000	DCX H	;Back down the pointer to the ALRMM3 buffer
	67E2	BE	6001	CMP M	;Check for Counter<Alarm
	67E3	DA 67F4	6002	JC ITRUN0	;Jump if it is
	67E6	1B	6003	DCX D	;Back up the ITTEMP pointer
	67E7	2B	6004	DCX H	;and the ALRMM3 pointer
	67E8	1A	6005	LDAX D	;Get the next counter byte
	67E9	BE	6006	CMP M	;Check for Counter<Alarm
	67EA	D2 67F4	6007	JNC ITRUN0	;Jump if it is
	67ED	1B	6008	DCX D	;Back up the ITTEMP pointer
	67EE	2B	6009	DCX H	;and the ALRMM3 pointer
	67EF	1A	6010	LDAX D	;Get the next counter byte
	67F0	BE	6011	CMP M	;Check for Counter<Alarm
	67F1	D2 67F4	6012	JNC ITRUN0	;Jump if it is
			6013		
			6014	;handle the special case here	
			6015		
	67F4	D3 2D	6016	ITRUN0: OUT SETTMR	;Start up the counter
	67F6	F3	6017	ITRUN2: DI	;Disable interrupts for a moment
	67F7	20	6018	RIM	;Get the interrupt masks
	67F8	E6 FE	6019	ANI 0FEH	;Clear M5.5
	67FA	F6 08	6020	ORI 8H	;Set MSE
	67FC	30	6021	SIM	;Write the masks
	67FD	FB	6022	EI	;Enable interrupts
	67FE	C9	6023	RET	;Leave
			6024		
			6025		
	67FF	D3 2C	6026	ITRUN1: OUT CLRTMR	;Turn the timer off
	6801	CD 6823	6027	CALL XFRSET	;Go load the Alarm registers
	6804	AF	6028	XRA A	;Make a 0

Error	Addr	Code	Seq	Source statement	
6805	32	7099	6029	STA IMPXFR	;Clear the Implied XFR flag
6808	3C		6030	INR A	;Make a 1
6809	32	7098	6031	STA BLKFLG	;Set the flag to indicate the 100Hz is blocked
680C	D3	3B	6032	OUT SETBLK	;Block the 100Hz clock
680E	3E	63	6033	MVI A,LARM12	;Prepare to Load and Arm counters 1 and 2
6810	D3	1D	6034	OUT ITCSR	;Do it
6812	AF		6035	XRA A	;Make a 0
6813	32	7098	6036	STA BLKFLG	;Zero the 100Hz clock blocked flag
6816	D3	3A	6037	OUT CLRBLK	;Unblock the 100Hz clock
6818	D3	2D	6038	OUT SETTMR	;Start the counter
681A	3E	40	6039	MVI A,040H	;Set up code to drive SOD low to clear any
681C	30		6040	SIM	;pending Timer interrupt.
681D	3E	C0	6041	MVI A,0C0H	;Code to drive SOD high
681F	30		6042	SIM	
6820	C3	67F6	6043	JMP ITRUN2	;Go clear M5.5
			6044		
			6045		
6823	21	7083	6046	XFRSET: LXI H,NICFLG	;Point to the new NIC value flag
6826	7E		6047	MOV A,M	;Get it
6827	A7		6048	ANA A	;Set or clear?
6828	CA	684C	6049	JZ NEWCNT	;Skip the next part if no new NICR
682B	35		6050	XFRSE0: DCR M	;Otherwise, clear it now
682C	2A	7084	6051	LHLD NICR	;Get two bytes
682F	22	7090	6052	SHLD ALARM	;Move them over
6832	2A	7086	6053	LHLD NICR+2	;Two more
6835	22	7092	6054	SHLD ALARM+2	;and so on
6838	2A	7088	6055	LHLD NICR+4	
683B	22	7094	6056	SHLD ALARM+4	
683E	2A	708A	6057	LHLD NICR+6	
6841	22	7096	6058	SHLD ALARM+6	
6844	21	7080	6059	LXI H,SMLNIC	;Get the Small NICR flag
6847	7E		6060	MOV A,M	;Put it in the AC
6848	36	00	6061	MVI M,0	;Zero it out in memory
684A	23		6062	INX H	;Point to the Small Alarm flag
684B	77		6063	MOV M,A	;Set or clear it with the Small NICR flag
684C	21	7082	6064	NEWCNT: LXI H,LOADED	;Point to the 'which count is loaded' flag byte
684F	11	7090	6065	LXI D,ALARM	;Point to the unaltered count
6852	06	00	6066	MVI B,0	;Set up a 0 in B
6854	3A	707C	6067	LDA ICCS	;Get the current version of the ICCS
6857	1F		6068	RAR	;Check the Run bit
6858	D2	6878	6069	JNC NEWCN1	;Jump if clear
685B	11	7094	6070	LXI D,ALRM3	;Point to the minus 3 version of the count
685E	04		6071	INR B	;Make a 1 in B
685F	3A	7081	6072	LDA SMLALM	;Get the Small Alarm flag
6862	A7		6073	ANA A	;Is it set?
6863	CA	6878	6074	JZ NEWCN1	;Jump if clear
6866	3A	7083	6075	LDA NICFLG	;Since the value to go in the Alarm registers is
6869	A7		6076	ANA A	;too small, we must see if there is a backup NIC
686A	CA	6874	6077	JZ NEWCNO	;Jump if there isn't
686D	3A	7080	6078	LDA SMLNIC	;See if the backup NIC is too small

Error	Addr	Code	Seq	Source statement	
6870	A7		6079	ANA A	
6871	CA 6878		6080	JZ NEWCN1	;Jump if it isn't
6874	11 708C		6081	NEWCN0: LXI D,MINCNT	;Point to the minimum value of the count
6877	04		6082	INR B	;Make a 2 in B
6878	70		6083	NEWCN1: MOV M,B	;Store the correct flags in LOADED
6879	EB		6084	XCHG	;Put the correct pointer in HL
687A	3E 01		6085	NEWCN2: MVI A,1	;Make a 1
687C	32 7098		6086	STA BLKFLG	;Set the flag to indicate the 100Hz is blocked
687F	D3 3B		6087	OUT SETBLK	;Block the 100Hz clock
6881	3E 07		6088	MVI A,ALARM1	;Point to the Alarm registers
6883	D3 1D		6089	OUT ITCSR	;Do it
6885	7E		6090	MOV A,M	;Get the low byte
6886	23		6091	INX H	;Increment the pointer
6887	D3 1C		6092	OUT ITDB	;Write it to the Alarm reg.
6889	7E		6093	MOV A,M	;Get the next byte
688A	23		6094	INX H	;Bump the pointer
688B	D3 1C		6095	OUT ITDB	;Write it
688D	7E		6096	MOV A,M	;Get the next byte
688E	23		6097	INX H	;Bump the pointer
688F	D3 1C		6098	OUT ITDB	;Write the byte
6891	7E		6099	MOV A,M	;Get the last byte
6892	D3 1C		6100	OUT ITDB	;and write it
6894	AF		6101	XRA A	;Make a 0
6895	32 7098		6102	STA BLKFLG	;Zero the 100Hz clock blocked flag
6898	D3 3A		6103	OUT CLRBLK	;Unblock the 100Hz clock
689A	C9		6104	RET	;Leave
			6105		
			6106		
			6107		
689B	3A 707C		6108	ITRCSR: LDA ICCS	;Get the ICCS
689E	47		6109	MOV B,A	;Put a copy in B
689F	1F		6110	RAR	;Is the run bit on?
68A0	78		6111	MOV A,B	;Put the ICCS in A just in case Run=0
68A1	D2 6590		6112	JNC LAST	;Go send this version of the ICCS to the CPU
68A4	3A 7082		6113	LDA LOADED	;Find out what is in the Alarm registers
68A7	FE 02		6114	CPI 2	;If it is the Minimum count we want to indicate
			6115		;an Overrun Error
68A9	78		6116	MOV A,B	;Get the ICCS into A
68AA	C2 6590		6117	JNZ LAST	;Jump if not the Minimum count
68AD	F6 60		6118	ORI 60H	;Set the Overrun Error and INT bits
68AF	C3 6590		6119	JMP LAST	;Go send the ICCS byte
			6120		
			6121		
68B2	3A 7082		6122	ITICR: LDA LOADED	;Get the Loaded Flag byte
68B5	FE 02		6123	CPI 2	;Look for minimum value loaded
68B7	C2 68C1		6124	JNZ ITICR0	;Jump if not
68BA	3A 707C		6125	LDA ICCS	;Get the ICCS
68BD	1F		6126	RAR	;Is the timer running?
68BE	DA 68FD		6127	JC ITICR2	;Jump if it is
68C1	3E 01		6128	ITICR0: MVI A,1	;Make a 1

Error	Addr	Code	Seq	Source statement	
68C3	32	7098	6129	STA	BLKFLG ;Set the flag to indicate the 100Hz is blocked
68C6	D3	3B	6130	OUT	SETBLK ;Block the 100Hz clock
68C8	3E	A3	6131	MVI	A,SAVE12 ;Get the current value of the count
68CA	D3	1D	6132	OUT	ITCSR ;Now
68CC	3E	19	6133	MVI	A,HOLD1 ;Point the pointer to the Hold registers
68CE	D3	1D	6134	ITICR1: OUT	ITCSR ;Now
68D0	AF		6135	XRA	A ;Make a 0
68D1	32	7098	6136	STA	BLKFLG ;Zero the 100Hz clock blocked flag
68D4	D3	3A	6137	OUT	CLRBLK ;Unblock the 100Hz clock
68D6	DB	1C	6138	IN	ITDB ;Get the low byte
68D8	D3	CC	6139	OUT	WRITE ;Send it
68DA	CD	6971	6140	CALL	WAITH ;
68DD	DB	1C	6141	IN	ITDB ;Get the next byte
68DF	D3	CC	6142	OUT	WRITE ;Send it to the CPU
68E1	CD	6980	6143	CALL	WAITH ;
68E4	DB	1C	6144	IN	ITDB ;Next byte
68E6	D3	CC	6145	OUT	WRITE ;Send it
68E8	CD	6971	6146	CALL	WAITH ;
68EB	DB	1C	6147	IN	ITDB ;Get the last byte
68ED	D3	CC	6148	ITICR3: OUT	WRITE ;Send it
68EF	CD	6980	6149	CALL	WAITH ;
68F2	21	7099	6150	LXI	H,IMPXFR ;Get this byte
68F5	7E		6151	MOV	A,M ;
68F6	D3	CC	6152	OUT	WRITE ;Send it to the CPU micro engine
68F8	D3	A9	6153	OUT	CLRACK ;Clear Ack
68FA	36	00	6154	MVI	M,0 ;Zero my copy of IMPXFR
68FC	C9		6155	RET	;Leave
			6156		
			6157		;The following section is used when the count the program was trying to use
			6158		;was too small to be allowed. If the ICR is read under this circumstance it
			6159		;will always read 0. If the ICCS is read it will always have the INT and
			6160		;OVERRUN bits on.
			6161		
			6162		
68FD	21	7090	6163	ITICR2: LXI	H,ALARM ;Get the unaltered count
6900	7E		6164	MOV	A,M ;and send it to the CPU
6901	D3	CC	6165	OUT	WRITE ;Send it
6903	CD	6971	6166	CALL	WAITH ;
6906	23		6167	INX	H ;Point to the next byte
6907	7E		6168	MOV	A,M ;Get it
6908	DB	1C	6169	IN	ITDB ;Get the next byte
690A	D3	CC	6170	OUT	WRITE ;Send it to the CPU
690C	CD	6980	6171	CALL	WAITH ;
690F	23		6172	INX	H ;Point to the next byte
6910	7E		6173	MOV	A,M ;Get it
6911	D3	CC	6174	OUT	WRITE ;Send it
6913	CD	6971	6175	CALL	WAITH ;
6916	23		6176	INX	H ;Point to the last byte
6917	7E		6177	MOV	A,M ;Get it
6918	C3	68ED	6178	JMP	ITICR3 ;Go share the code that sends the last byte

Error	Addr	Code	Seq	Source statement	
			6179		;and IMPXFR
			6180		
			6181		
691B	21	7084	6182	ITNICR: LXI H,NICR	;Point to the storage
691E	CD	6953	6183	CALL NICSUB	;The next part is a subroutine so TODW can
			6184		;also use it
6921	21	7088	6185	LXI H,NICRM3	;Point to the dest
6924	11	7084	6186	LXI D,NICR	;and the source
6927	CD	00F9	6187	CALL COPY	;Move it over
692A	21	708B	6188	LXI H,NICRM3+3	;Point to the high byte of the minus 3 version
692D	7E		6189	MOV A,M	;Get it
692E	2B		6190	DCX H	;Back the pointer down
692F	B6		6191	ORA M	;OR in the next byte down
6930	2B		6192	DCX H	;Back the pointer down again
6931	B6		6193	ORA M	;OR in the next one
6932	2B		6194	DCX H	;Point to the low byte
6933	C2	6947	6195	JNZ ITNICO	;Jump if high three bytes not 0
6936	7E		6196	MOV A,M	;Get the low byte
6937	FE	E8	6197	CPI 232	;See if it is less min+2
6939	D2	6947	6198	JNC ITNICO	;Jump if not
693C	A7		6199	ANA A	;Is it 0 (the largest count)
693D	CA	6947	6200	JZ ITNICO	;Jump if it is
6940	CD	694D	6201	CALL ITNIC1	;Go set the NIC flag
6943	32	7080	6202	STA SHLNIC	;Set the Small NICR flag
6946	C9		6203	RET	;Leave
			6204		
6947	01	719E	6205	ITNICO: LXI B,NEG3	;Point to a minus 3
694A	CD	6251	6206	CALL ADDAD1	;Go add it in
694D	3E	01	6207	ITNIC1: MVI A,01	;Get a one into A
694F	32	7083	6208	STA NICFLG	;Set the NICR Flag
6952	C9		6209	RET	;All done
			6210		;use it
6953	DB	80	6211	NICSUB: IN READ	;Get the low byte
6955	77		6212	MOV M,A	;Store it
6956	23		6213	INX H	;Bump the pointer
6957	CD	6971	6214	CALL WAITH	;Go wait for next byte
695A	DB	80	6215	IN READ	;Get next byte
695C	77		6216	MOV M,A	;Store it
695D	23		6217	INX H	;Bump the pointer
695E	CD	6980	6218	CALL WAITL	;Go wait
6961	DB	80	6219	IN READ	;Get the next byte
6963	77		6220	MOV M,A	;Store it
6964	23		6221	INX H	;Bump the pointer
6965	CD	6971	6222	CALL WAITH	;Wait for CPATTN to go high
6968	DB	80	6223	IN READ	;Get the last byte
696A	77		6224	MOV M,A	;And store it
696B	CD	6980	6225	CALL WAITL	;Wait for CPATTN to go away
696E	D3	A9	6226	OUT CLRAK	;Clear Console Ack
6970	C9		6227	RET	;Leave
			6228		

Error	Addr	Code	Seq	Source statement
-------	------	------	-----	------------------

			6229	
			6230	
6971	D3	A9	6231	WAITH: OUT CLRACK ;Get rid of Ack
6973	1E	00	6232	MVI E,0 ;Prime the count
6975	DB	83	6233	1\$: IN CPATTN ;Get CPATTN
6977	1F		6234	RAR ;Check to see if it is set
6978	D8		6235	RC ;We're done if it is
6979	1C		6236	INR E ;Bump the count
697A	C2	6975	6237	JNZ 1\$;Loop if not zero
697D	CD	5D5E	6238	CALL COMERR ;Else timeout
			6239	
			6240	
			6241	
			6242	
6980	D3	A8	6243	WAITL: OUT SETACK ;Set Console Ack
6982	1E	00	6244	MVI E,0 ;Prime the count
6984	DB	83	6245	1\$: IN CPATTN ;Get CPATTN
6986	1F		6246	RAR ;Is it clear?
6987	D0		6247	RNC ;Return if it is
6988	1C		6248	INR E ;Else bump the count
6989	C2	6984	6249	JNZ 1\$;Loop if count doesn't equal 0
698C	CD	5D5E	6250	CALL COMERR ;Else timeout
			6251	
			6252	
			6253	
			6254	
			6255	
			6256	;SAVREG IS ENTERED FROM THE HALT FLOWS. IT ALLOWS FOR CONTINUE AND
			6257	;SI TO OCCUR
			6258	
			6259	
			6260	
			6261	
			6262	
			6263	
			6264	
698F	D3	A0	6265	SETPAR: OUT ENBPARG ;Turn on Parity
6991	C3	4C32	6266	JMP IDLE ;Go back to the Null Task
			6267	
6994	D3	A1	6268	CLRPAR: OUT DISPAR ;Turn off parity
6996	C3	4C32	6269	JMP IDLE ;Go back to Null Task
			6270	
			6271	
6999	D3	2C	6272	TMRPRP: OUT CLRTMR ;Make sure the counter isn't counting
699B	D3	A5	6273	OUT CLRTIM ;Clear the Timer interrupt to the CPU
699D	21	708C	6274	LXI H,MINCNT ;Point to the minimum allowable count
69A0	CD	687A	6275	CALL NEWCN2 ;Load the alarm registers with it
69A3	3E	01	6276	MVI A,1 ;Make a 1
69A5	32	7098	6277	STA BLKFLG ;Set the flag to indicate the 100Hz is blocked
69A8	D3	3B	6278	OUT SETBLK ;Block the 100Hz clock

Error	Addr	Code	Seq	Source statement	
	69AA	3E 17	6279	MVI A,MASTER	;Code to force internal pointer to MM reg
	69AC	D3 1D	6280	OUT ITCSR	;Send it out
	69AE	AF	6281	XRA A	;Make a 0
	69AF	32 7098	6282	STA BLKFLG	;Zero the 100Hz clock blocked flag
	69B2	D3 3A	6283	OUT CLRBLK	;Unblock the 100Hz clock
	69B4	DB 1C	6284	IN ITDB	;Get the low byte of the MM reg
	69B6	FE 0C	6285	CPI 0CH	;See if the low byte is correct
	69B8	C2 69BF	6286	JNZ TMRPRO	;Jump if not
	69BB	DB 1C	6287	IN ITDB	;Get the high byte
	69BD	FE 0A	6288	CPI 0AH	;See if it is correct
	69BF	C4 69DE	6289	TMRPRO: CNZ TPSUB	;Go init the 9513 if not
	69C2	3E 01	6290	MVI A,1	;xMove 1 to A
	69C4	32 7098	6291	STA BLKFLG	;xSet the flag to indicate that 100Hz is blocked
	69C7	D3 3B	6292	OUT SETBLK	;xBlock the 100Hz clock
	69C9	3E C4	6293	MVI A,0C4H	;Disarm counter #3
	69CB	D3 1D	6294	OUT ITCSR	;Send it out
	69CD	3E E3	6295	MVI A,0E3H	;Code for setting output of counter #3 high
	69CF	D3 1D	6296	OUT ITCSR	;Send it out to the chip
	69D1	AF	6297	XRA A	;xMake A 0
	69D2	32 7098	6298	STA BLKFLG	;xSet the flag to indicate the 100Hz is unblocked
	69D5	D3 3A	6299	OUT CLRBLK	;xUnblock the 100hz clock
	69D7	3E 40	6300	TMRPR1: MVI A,040H	;Reset any garbage 9513 overflow interrupts
	69D9	30	6301	SIM	;to the 8085 by driving SOD low then high
	69DA	3E C0	6302	MVI A,0C0H	;
	69DC	30	6303	SIM	;
	69DD	C9	6304	RET	;Leave
			6305		
	69DE	3E 01	6306	TPSUB: MVI A,1	;Make a 1
	69E0	32 7098	6307	STA BLKFLG	;Set the flag to indicate the 100Hz is blocked
	69E3	D3 3B	6308	OUT SETBLK	;Block the 100Hz clock
	69E5	3E 5F	6309	MVI A,05FH	;xThis should be part of init routine
	69E7	D3 1D	6310	OUT ITCSR	;xThis should do software retrigger
	69E9	3E 01	6311	MVI A,CG1MOD	;Point the internal pointer to Counter
	69EB	D3 1D	6312	OUT ITCSR	;Group 1, Mode register
	69ED	21 7052	6313	LXI H,TMRFIL	;Point to the Timer Prep File
	69F0	06 05	6314	MVI B,5	;Overall loop count
	69F2	0E 06	6315	TPSUB0: MVI C,6	;Inner loop count
	69F4	7E	6316	TPSUB1: MOV A,H	;Get Counter Mode low byte
	69F5	D3 1C	6317	OUT ITDB	;Send it out
	69F7	23	6318	INX H	;Bump the Pointer
	69F8	0D	6319	DCR C	;Reduce the inner count
	69F9	C2 69F4	6320	JNZ TPSUB1	;Stay in the inner loop if not 0
	69FC	05	6321	DCR B	;Decrement the group count
	69FD	C2 69F2	6322	JNZ TPSUB0	;Loop if all the groups aren't done
	6A00	3E 17	6323	MVI A,MASTER	;Point to the Master Mode reg
	6A02	D3 1D	6324	OUT ITCSR	;Do it
	6A04	21 0A0C	6325	LXI H,MSTRCD	;Put the code for the Master Mode reg into HL
	6A07	7D	6326	MOV A,L	;Get the low byte
	6A08	D3 1C	6327	OUT ITDB	;Send it
	6A0A	7C	6328	MOV A,H	;Get the High byte

Error	Addr	Code	Seq	Source statement	
	6A0B	D3 1C	6329	OUT ITDB	;Send it
	6A0D	3E 7B	6330	MVI A,LA1245	;Load and Arm counters 1, 2, 4 and 5
	6A0F	D3 1D	6331	OUT ITCSR	;1 and 2 won't count because gate is off.
	6A11	AF	6332	XRA A	;Make a 0
	6A12	32 709B	6333	STA BLKFLG	;Zero the 100Hz clock blocked flag
	6A15	D3 3A	6334	OUT CLRBLK	;Unblock the 100Hz clock
	6A17	C9	6335	RET	;Leave
			6336		
			6337	;	
			6338	;	
	6A18	F5	6339	TMRSRV: PUSH PSW	;Save the AC and the PSW
	6A19	E5	6340	PUSH H	;Save H and L
	6A1A	20	6341	RIM	;Get the interrupt masks
	6A1B	F6 09	6342	ORI 9H	;Set the MSE and MS.5
	6A1D	30	6343	SIM	;Write them out
	6A1E	FB	6344	EI	;Turn on the interrupt system again
	6A1F	3A 707C	6345	LDA ICCS	;Get the ICCS
	6A22	17	6346	RAL	;Check the I.E.
	6A23	D2 6A2B	6347	JNC TMRSR0	;Jump if no I.E.
	6A26	D3 A4	6348	OUT SETTIM	;Set the Timer interrupt to the CPU
	6A28	3E 01	6349	TMRSR0: MVI A,1	;Make a 1
	6A2A	32 709B	6350	STA BLKFLG	;Set the flag to indicate the 100Hz is blocked
	6A2D	D3 3B	6351	OUT SETBLK	;Block the 100Hz clock
	6A2F	3E 62	6352	MVI A,LARM2	;Prepare to Load and Arm counter 2
	6A31	D3 1D	6353	OUT ITCSR	;Do it
	6A33	AF	6354	XRA A	;Make a 0
	6A34	32 709B	6355	STA BLKFLG	;Zero the 100Hz clock blocked flag
	6A37	D3 3A	6356	OUT CLRBLK	;Unblock the 100Hz clock
	6A39	21 7083	6357	LXI H,NICFLG	;Point to this flag
	6A3C	7E	6358	MOV A,M	;Get it
	6A3D	A7	6359	ANA A	;Is the NIC flag set?
	6A3E	CA 6A4B	6360	JZ TMRSR1	;Jump if not
	6A41	32 7099	6361	STA IMPXFR	;Set the 'Implied XFR occurred' flag
	6A44	C5	6362	PUSH B	;Save BC
	6A45	D5	6363	PUSH D	;and DE
	6A46	CD 682B	6364	CALL XFRSE0	;Go figure out what to load and load it
	6A49	D1	6365	POP D	;Get DE back
	6A4A	C1	6366	POP B	;and BC
	6A4B	21 707C	6367	TMRSR1: LXI H,ICCS	;Point to the ICCS
	6A4E	7E	6368	MOV A,M	;Get the ICCS
	6A4F	E6 40	6369	ANI 040H	;Check for INT already set
	6A51	CA 6A55	6370	JZ TMRSR2	;Jump if not already set
	6A54	1F	6371	RAR	;This will set bit 05 quickly
	6A55	F6 40	6372	TMRSR2: ORI 040H	;Set the INT bit
	6A57	B6	6373	ORA M	;OR in the ICCS
	6A58	77	6374	MOV M,A	;Store the new ICCS
	6A59	1F	6375	RAR	;Is the Run bit on?
	6A5A	D2 6A6E	6376	JNC TMRSR4	;Jump if it is off
	6A5D	3E 40	6377	MVI A,040H	;Set SOE and clear SOD
	6A5F	30	6378	SIM	;Drive SOD low, then high to reset Timer Int

Error	Addr	Code	Seq	Source statement	
	6A60	3E C0	6379	MVI A,0C0H	:
	6A62	30	6380	SIM	:
	6A63	F3	6381	DI	;Turn off the interrupt system
	6A64	20	6382	RIM	;Get the interrupt masks
	6A65	E6 FE	6383	ANI 0FEH	;Clear MS.5
	6A67	F6 08	6384	ORI 8H	;Set MSE
	6A69	30	6385	SIM	;Write the masks
	6A6A	E1	6386	TMRSR3: POP H	;Get HL back
	6A6B	F1	6387	POP PSW	;Get the PSW and the AC back
	6A6C	FB	6388	EI	;Turn on the interrupts
	6A6D	C9	6389	RET	;Leave
			6390		
	6A6E	D3 2C	6391	TMRSR4: OUT CLRTMR	;Turn off the timer
	6A70	3E 63	6392	MVI A,LARM12	;Put the value in A
	6A72	D3 1D	6393	OUT ITCSR	;Load and arm counters 1 and 2
	6A74	3E 40	6394	MVI A,040H	;Set SOE and clear SOD
	6A76	30	6395	SIM	;Drive SOD low, then high to reset Timer Int
	6A77	3E C0	6396	MVI A,0C0H	:
	6A79	30	6397	SIM	:
	6A7A	C3 6A6A	6398	JMP TMRSR3	;Go clean things up
			6399	:	
			6400	;	
			6401		
			6402		
			6403		
	6A7D	3E 01	6404	TODR: MVI A,1	;Make a 1
	6A7F	32 7098	6405	STA BLKFLG	;Set the flag to indicate the 100Hz is blocked
	6A82	D3 3B	6406	OUT SETBLK	;Block the 100Hz clock
	6A84	3E B8	6407	MVI A,SAVE45	;Save Counters 4 and 5
	6A86	D3 1D	6408	OUT ITCSR	;Do it
	6A88	3E 1C	6409	MVI A,HOLD4	;Point at the Hold registers
	6A8A	C3 68CE	6410	JMP ITICR1	;This will do the rest for us
			6411		
	6A8D	21 709A	6412	TODW: LXI H,TODBFR	;Put the value here at first
	6A90	CD 6953	6413	CALL NICSUB	;This code will get the value for us
	6A93	21 709A	6414	TODW1: LXI H,TODBFR	;Point to the value to be loaded
	6A96	3E 01	6415	MVI A,1	;Make a 1
	6A98	32 7098	6416	STA BLKFLG	;Set the flag to indicate the 100Hz is blocked
	6A9B	D3 3B	6417	OUT SETBLK	;Block the 100Hz clock
	6A9D	CD 6AB5	6418	CALL TODPRP	;go put it in
	6AA0	3E 58	6419	MVI A,LOAD45	;Load Counters 4 and 5
	6AA2	D3 1D	6420	OUT ITCSR	;Now
	6AA4	21 70FD	6421	LXI H,ZEROS	;Point to some zeros
	6AA7	CD 6AB5	6422	CALL TODPRP	;And reload the load registers
	6AAA	3E 38	6423	MVI A,ARM45	;Start counters 4 and 5 counting
	6AAC	D3 1D	6424	OUT ITCSR	;Do it
	6AAE	AF	6425	XRA A	;Make a 0
	6AAF	32 7098	6426	STA BLKFLG	;Zero the 100Hz clock blocked flag
	6AB2	D3 3A	6427	OUT CLRBLK	;Unblock the 100Hz clock
	6AB4	C9	6428	RET	;And leave

Error	Addr	Code	Seq	Source statement
			6429	
			6430	;.....
			6431	;
			6432	;The TODPRP subroutine expects the calling routine to block the 100Hz clock
			6433	;
6AB5	3E	0C	6434	TODPRP: MVI A,CGFOUR ;Force the 9513 Internal Pointer to counter 4
6AB7	D3	1D	6435	OUT ITCSR ;Do it
			6436	;Calling routine must have loaded HL
6AB9	7E		6437	MOV A,M ;Get the low byte
6ABA	D3	1C	6438	OUT ITDB ;Send it
6ABC	23		6439	INX H ;Point to the next byte
6ABD	7E		6440	MOV A,M ;Get it
6ABE	D3	1C	6441	OUT ITDB ;and send it
6AC0	3E	0D	6442	MVI A,CGFIVE ;Point to counter five
6AC2	D3	1D	6443	OUT ITCSR ;Do it
6AC4	23		6444	INX H ;Point to the next byte
6AC5	7E		6445	MOV A,M ;and get it
6AC6	D3	1C	6446	OUT ITDB ;and send it
6AC8	23		6447	INX H ;Point to the last byte
6AC9	7E		6448	MOV A,M ;and get it
6ACA	D3	1C	6449	OUT ITDB ;and send it
6ACC	C9		6450	RET ;Leave
			6451	;
			6452	;.....
			6453	
			6454	
6ACD	D3	35	6455	UBINIT: OUT INITH ;Set Unibus Init
6ACF	CD	5760	6456	CALL STAL10 ;Go stall for 10 Ms
6AD2	D3	34	6457	OUT INITL ;Release Unibus Init
6AD4	D3	A9	6458	OUT CLRACK ;Clear Console Ack
6AD6	C9		6459	RET ;And leave
			6460	
6AD7	D3	A9	6461	SOFDON: OUT CLRACK ;Turn the CPU loose
6AD9	C9		6462	RET ;
			6463	
6ADA	D3	A9	6464	PMBOOT: OUT CLRACK ;Turn the CPU loose
6ADC	D3	20	6465	OUT CLRCLK ;Stop the CPU
6ADE	D3	3C	6466	OUT CLRLIT ;Turn off the Run Light
6AE0	AF		6467	XRA A ;Make a 0
6AE1	32	40D3	6468	STA RUNFLG ;Turn off this flag
6AE4	C3	6020	6469	JMP CLDSTX ;Go try to boot the system up
			6470	
6AE7	D3	A9	6471	CLRCLD: OUT CLRACK ;Turn the CPU loose
6AE9	AF		6472	XRA A ;Make a 0
6AEA	32	408A	6473	STA COLD ;Reset the Cold Start in Progress flag
6AED	32	40B6	6474	STA PUBFLG ;Clear the Power Up Boot Flag
6AF0	C9		6475	RET ;leave
			6476	
6AF1	D3	A9	6477	CLRWRM: OUT CLRACK ;Turn the CPU loose
6AF3	AF		6478	XRA A ;Make a 0

Error	Addr	Code	Seq	Source statement
	6AF4	32 41B7	6479	STA WARM ;Reset the Warm Start in Progress flag
	6AF7	C9	6480	RET ;Leave
			6481	
	6AF8	C9	6482	CLRPME: RET ;
			6483	
	6AF9	C9	6484	SETPME: RET ;
			6485	
			6486	
	6AFA	16 04	6487	ZROBUF: MVI D,04H ;Set up the count
	6AFC	AF	6488	ZBUF1: XRA A ;Generate 0
	6AFD	E5	6489	ZBUF2: PUSH H ;Save the address
	6AFE	77	6490	1\$: MOV H,A ;Clear the location
	6AFF	23	6491	INX H ;Bump the address
	6B00	15	6492	DCR D ;Decrement the count
	6B01	C2 6AFE	6493	JNZ 1\$;Loop if not done
	6B04	E1	6494	POP H ;Restore HL
	6B05	C9	6495	RET ;Leave
			6496	;
			6497	;.....
			6498	
			6499	
			6500	
			6501	
			6502	
			6503	
			6504	
			6505	
			6506	;.....
			6507	;.....
			6508	;.....
			6509	;
			6510	;
			6511	; Many of the buffers and constants used by the console software
			6512	; (not MICRON) start here. The vectors for the Program Mode command decode
			6513	; and the buffer for the Console Mode indirect command files are here as well
			6514	;
			6515	
			6516	
	=7000		6517	ORG 07000H ;
			6518	
7000	00		6519	CHRCNT: DB 0
7001	=0050		6520	CHRBUF: DS 080
			6521	
7051	00		6522	TIMCNT: DB 0
			6523	
7052	A2A9		6524	TMRFIL: DW 0A2A9H ;
7054	0000 0000		6525	DW 0,0 ;
			6526	
7058	A0A9		6527	DW 0A0A9H ;
705A	0000 0000		6528	DW 0,0 ;

Error	Addr	Code	Seq	Source statement
			6529	
705E	020E		6530	DW 20EH
7060	F830 0000		6531	DW 0F830H,0
			6532	;Count for Power Fail Timeout
7064	0128		6533	DW 128H
7066	0000 0000		6534	DW 0.0
			6535	
706A	0028		6536	DW 28H
706C	0000 0000		6537	DW 0.0
			6538	
7070	00		6539	ISR: DB 0
7071	00		6540	IPR: DB 0
			6541	
7072	00		6542	RCVCSR: DB 0
7073	00		6543	RCVDB: DB 0
			6544	
7074	00		6545	INTCSR: DB 0
7075	00		6546	INTDB: DB 0
7076	00		6547	XMTCSR: DB 0
7077	00		6548	XMTDB: DB 0
			6549	
7078	0000		6550	XCSADR: DW 0
707A	0000		6551	XDBADR: DW 0
			6552	
707C	00		6553	ICCS: DB 0
707D	00		6554	OLDRUN: DB 0
707E	00		6555	ITSGL: DB 0
			6556	
			6557	
707F	00		6558	HMPEND: DB 0
			6559	
			6560	
			6561	;****
			6562	
7080	00		6563	SMLNIC: DB 0
			6564	
7081	00		6565	SMLALM: DB 0
			6566	
			6567	
			6568	;****
			6569	
			6570	
7082	00		6571	LOADED: DB 0
			6572	
			6573	
			6574	
7083	00		6575	NICFLG: DB 0
			6576	
7084	=0004		6577	NICR: DS 4
7088	=0004		6578	NICRM3: DS 4

Error	Addr	Code	Seq	Source statement
-------	------	------	-----	------------------

708C	00E8	6579	MINCNT: DW	232 ;Minimum allowable count
708E	0000	6580	DW	0 ;

7090	=0004	6581		
7094	=0004	6582	ALARM: DS	4 ;
		6583	ALRMM3: DS	4 ;

7098	00	6584		
		6585	BLKFLG: DB	0 ;100Hz blocked flag

7099	00	6586		
		6587	IMPXFR: DB	0 ;

709A	=0004	6588		
		6589	TODBFR: DS	4

709E	=0004	6590		
		6591	ITTEMP: DS	4 ;

6592
6593 ;
6594 ;
6595 ;
6596 ; This is a table of addresses used by Program Mode to determine which
6597 ; command the CPU wants to execute. The CPU passed the Program Mode software
6598 ; a command byte which is actually used as an offset into this table.
6599 ;

70A2	C3 6575	6600	TABLE: JMP	INTACK
70A5	C3 658C	6601	JMP	TTXCSR
70A8	C3 6595	6602	JMP	TTRCSR
70AB	C3 65A0	6603	JMP	TTXDB
70AE	C3 65BB	6604	JMP	TTXIE
70B1	C3 65C2	6605	JMP	TTRIE
70B4	C3 65FB	6606	JMP	TTXID
70B7	C3 6606	6607	JMP	TTRID
70BA	C3 6625	6608	JMP	PMHALT
70BD	C3 6656	6609	JMP	TTRDB

70C0	C3 659B	6610		
70C3	C3 6580	6611	JMP	TURCSR
70C6	C3 6683	6612	JMP	TUXDB
70C9	C3 65C9	6613	JMP	TURDB
70CC	C9	6614	JMP	TUWCSR

70CD	00	6615	RET	;Dummy this out
70CE	00	6616	NOP	;
70CF	C3 65F4	6617	NOP	;

70D2	C3 661A	6618	JMP	TURIE
		6619	JMP	TURID

70D5	C3 689B	6620		
70D8	C3 66C6	6621	JMP	ITRCSR
70DB	C3 6882	6622	JMP	ITWCSR
70DE	C3 691B	6623	JMP	ITICR
		6624	JMP	ITNICR

70E1	C3 6A7D	6625		
70E4	C3 6A8D	6626	JMP	TODR
		6627	JMP	TODW

Error Addr	Code	Seq	Source statement
70E7	C3 6ACD	6629 6630 6631	JMP UBINIT ;Go init the Unibus
70EA	C3 6AD7	6632	JMP SOFDON ;
70ED	C3 6ADA	6633	JMP PMBOOT ;
70F0	C3 6AF1	6634	JMP CLRWRM ;Clear the Warm Start Flag
70F3	C3 6AE7	6635 6636	JMP CLRCLD ;Clear the Cold Start flag
70F6	C3 6AF8	6637	JMP CLRPME ;
70F9	C3 6AF9	6638 6639	JMP SETPME ;
		6640	;
		6641	;
		6642	End of the table
		6643	
70FC	00	6644 6645	PCOUNT: DB 0 ;
70FD	0000	6646 6647	ZEROS: DW 0
70FF	0000	6648 6649	DW 0
7101	12	6650 6651	COUNT1: DB 18
7102	40 42 43	6652	CMPAR1: DB ' BCDEHILMNIRSTWX;!'
7105	44 45 48		
7108	49 4C 4D		
710B	4E 52 53		
710E	54 57 58		
7111	3B 21		
7113	0D	6653	ADRES1: DB CR
7114	5A1B	6654	DW INDRCT ;Indirect Command File command
7116	576D	6655	DW BOOT
7118	5840	6656	DW CONT
711A	4FDE	6657	DW DDECOD
711C	50F2	6658	DW EXAMIN
711E	58E8	6659	DW HALT
7120	56E6	6660	DW INIT
7122	58FC	6661	DW LOAD
7124	5976	6662	DW MICSTP
7126	59D6	6663	DW STEP
7128	4D0E	6664	DW REPEAT
712A	57F2	6665	DW S_COMMAND ;
712C	5A3E	6666	DW TEST
712E	5AAA	6667	DW WAICMD ;
7130	5AD8	6668	DW XFER
7132	5D03	6669	DW CRCMD ;
7134	5D03	6670	DW CRCMD
7136	5D03	6671	DW CRCMD ;
		6672	
		6673	

Error	Addr	Code	Seq	Source statement
-------	------	------	-----	------------------

7138	0C		6674	
7139	42 57 4C		6675	COUNT2: DB 12 ;
713C	50 56 43		6676	CMPAR2: DB 'BWLPCVCMGIUSN'
713F	4D 47 49			
7142	55 53 4E			

7145	4E92		6677	ADRES2: DW BYTE
7147	4E97		6678	DW WORD
7149	4E9C		6679	DW LONG
714B	4E6C		6680	DW PHYS
714D	4E71		6681	DW VRTUAL
714F	4E8A		6682	DW WCS
7151	4E7B		6683	DW MREG
7153	4E76		6684	DW GPR
7155	4E80		6685	DW IREG
7157	4E85		6686	DW MICRO
7159	4EF1		6687	DW STRTAD
715B	4EA4		6688	DW NEXTAD

6689
 6690
 6691 ;Any changes to the next table must take into account the way it is used
 6692 ;by PARSE when the command is a Load command. The count is altered by the
 6693 ;PARSE code and only the first part of the table is used. This means that any
 6694 ;changes to this table must be done with this in mind. For example, if
 6695 ;something is to be added that must be recognized by all commands, it would have
 6696 ;to go in the first part of the CMPAR3 and ADRES3 areas and the count in PARSE
 6697 ;for Load commands would have to be changed from 5 to 6
 6698
 6699

715D	0C		6700	COUNT3: DB 12 ;
715E	0D		6701	CMPAR3: DB CR
715F	20		6702	DB SPACE
7160	2F 3B 21		6703	DB '/;!*-- *SPR'

716A	4FD6		6704	ADRES3: DW CRFLOW
716C	4E4C		6705	DW SPCFLW
716E	4E56		6706	DW SLASH
7170	4FCE		6707	DW SEMI ;
7172	4FCE		6708	DW SEMI ;
7174	4F91		6709	DW PLUS
7176	4F96		6710	DW MINUS
7178	4F9B		6711	DW ATSIGN
717A	4FA0		6712	DW ASTRSK
717C	4F7E		6713	DW STKFLW
717E	4F4B		6714	DW PXXX
7180	4F6D		6715	DW RN

7182	0000 0000		6716	
			6717	ZERO: DW 0,0 ;4 bytes of 0

Error	Addr	Code	Seq	Source statement
			6718	
7186	0001 0000		6719	POS1: DW 01H,0 ;
			6720	
718A	0002 0000		6721	POS2: DW 02H,0 ;
			6722	
718E	0004 0000		6723	POS4: DW 04H,0 ;
			6724	
7192	0200 0000		6725	POS200: DW 200H,0 ;
			6726	
7196	FFFF FFFF		6727	NEG1: DW 0FFFFH,0FFFFH ;
			6728	
719A	FFFF FFFF		6729	NEG2: DW 0FFFEH,0FFFFH ;
			6730	
719E	FFFD FFFF		6731	NEG3: DW 0FFFDH,0FFFFH ;
			6732	
71A2	FFFC FFFF		6733	NEG4: DW 0FFFC H,0FFFFH ;
			6734	
71A6	FE00 FFFF		6735	NEG200: DW 0FE00H,0FFFFH ;
			6736	
71AA	4100 0000		6737	RAMADR: DJ 4100H,0 ;
71AE	3F00 0000		6738	RAMCNT: DW 3F00H,0 ;
			6739	
71B2	00		6740	NEWDL: DB 0 ;
71B3	00		6741	NEWAS: DB 0 ;
			6742	
71B4	00		6743	DATLEN: DB 0 ;
71B5	00		6744	ADRSPC: DB 0 ;
			6745	
71B6	00		6746	EDPACK: DB 0
71B7	00		6747	EDMOD: DB 0
71B8	=0004		6748	EDADDR: DS 4
71BC	=0004		6749	EDDATA: DS 4
			6750	
71C0	00		6751	RCVBUF: DB 0
71C1	00		6752	RCVMOD: DB 0
71C2	0000 0000		6753	RCVADR: DW 0,0
71C6	0000 0000		6754	RCVDAT: DW 0,0
			6755	
			6756	
71CA	00 00		6757	SCNPAK: DB 0,0 ;
71CC	0000 0000		6758	SCNADR: DW 0,0 ;
			6759	
71D0	01 03		6760	DEPPAK: DB 1,3 ;Deposit code and GPR space qualifier
71D2	000A 0000		6761	DEPADR: DW 0AH,0 ;R11 address
71D6	0000 0000		6762	DEPDAT: DW 0,0 ;
			6763	
71DA	00		6764	HLTBUF: DB 0 ;
71DB	00		6765	HLTCOD: DB 0
71DC	0000 0000		6766	HALTPC: DW 0,0 ;
71E0	0000 0000		6767	HLTPSL: DW 0,0 ;

Error	Addr	Code	Seq	Source statement
-------	------	------	-----	------------------

	71E4	0000 0000	6768 6769 6770	RESADR: DW 0,0 ;
	71E8	0000 0000	6771 6772	LNCSAV: DW 0,0 ;
	71EC	0000 0000	6773 6774	LNCSUM: DW 0,0 ;
	71F0	01 00	6775	WFPACK: DB 1,0 ;
	71F2	0000 0000	6776	WFADR: DW 0,0 ;
	71F6	0000 0000	6777 6778	WFDATA: DW 0,0 ;
	71FA	00	6779 6780	WRMCNT: DB 0 ;
	71FB	02	6781 6782	INITPK: DB 02
	71FC	03	6783	CONPAK: DB 03
	71FD	00	6784 6785	CONMOD: DB 0
	71FE	=0004	6786	ABUFFER: DS 4
	7202	=0004	6787 6788 6789 6790	DBUFFER: DS 4
	7206	0000	6791	NSADBF: DW 0
	7208	=0004	6792 6793 6794	NBUFFER: DS 4
	720C	0000	6795	FNPTR: DW 0 ;Storage for Address of File Name Buffer
	720E	0000	6796	EXTPTR: DW 0 ;Storage for Address of Extension Buffer
	7210	00	6797	FNCNT: DB 0
	7211	00	6798	EXTCNT: DB 0
	7212	00	6799 6800 6801	DEVFLG: DB 0
	7213	00	6802	FLAGS: DB 0
	7214	00	6803	FLAGS1: DB 0
	7215	00	6804	FLAGS2: DB 0
	7216	00	6805	FLAGS3: DB 0
	7217	00	6806 6807	RFLAG: DB 0
	7218	00	6808	QUAL1: DB 0
	7219	0000	6809 6810	MRKBUF: DW 0 ;
	721B	00	6811 6812	INDCNT: DB 0 ;Storage for count
	721C	7600	6813 6814	INDADR: DW INDBUF ;Storage for address into Indirect command file
	721E	00	6815	STPFLG: DB 0 ;
	721F	00	6816 6817	MICFLG: DB 0 ;

Error	Addr	Code	Seq	Source statement
			6818	
7220	00		6819	BRKFLG: DB 0 ;Break in Progress Flag byte
7221	00		6820	BRKCNT: DB 0 ;
			6821	
7222	00		6822	BPHALT: DB 0 ;Break Point Halt Flag
			6823	
7223	00		6824	IFLAG: DB 0 ;
			6825	
7224	14		6826	OUTBUF: DB 20
7225	00		6827	OUTLET: DB 0
7226	20		6828	DB SPACE
7227	=0008		6829	OUTADR: DS 8
722F	20		6830	DB SPACE
7230	=0008		6831	OUTDAT: DS 8
7238	0D		6832	DB CR ;
			6833	
			6834	
7239	40 50 4F		6835	PWRFIL: DB ' POWER',0,'.CMD' ;
723C	57 45 52			
723F	00 2E 43			
7242	4D 44			
7244	0D 0A 00	6836	DB	CR,LF,0 ;
		6837		
7247	40 43 4F	6838	CODFIL: DB	' CODE00.CMD' ;
724A	44 45 30			
724D	30 2E 43			
7250	4D 44			
7252	0D 0A 00	6839	DB	CR,LF,0 ;
		6840		
7255	40 44 44	6841	DEFFIL: DB	' DD0:DEFB00.CMD' ;
7258	30 3A 44			
725B	45 46 42			
725E	4F 4F 2E			
7261	43 4D 44			
7264	0D 0A 00	6842	DB	CR,LF,0 ;
		6843		
7267	40 44 44	6844	FOOFIL: DB	' DD0:FOOB00.CMD' ;
726A	30 3A 46			
726D	4F 4F 42			
7270	4F 4F 2E			
7273	43 4D 44			
7276	0D 0A 00	6845	DB	CR,LF,0 ;
		6846		
7279	40 44 44	6847	CRAFIL: DB	' DD1:CRAB00.CMD' ;
727C	31 3A 43			
727F	52 41 42			
7282	4F 4F 2E			
7285	43 4D 44			
7288	0D 0A 00	6848	DB	CR,LF,0 ;
		6849		

ZZ	RA
A	AC
AC	AD
AD	AL
AL	AR
AR	AT
AT	BA
BA	BL
BL	BO
BO	BR
BR	BY
BY	C
C	CF
CF	CG
CG	CH
CH	CL
CL	CL
CL	CL
CL	CL
CL	CL
CL	CL
CL	CM
CM	CM
CM	CN
CN	CN
CN	CN
CN	CO
CO	CO
CO	CO
CO	CO
CO	CO
CO	CO
CO	CO
CO	CP
CP	CP
CP	CR
CR	CR
CR	CS
CS	DA
DA	DC
DC	DE
DE	DE
DE	DE

22
M
RA

DE
DI
DI
DI
DN
DP
ED
EN
EN
EX
EX
EX
FI
FI
FL
FN
FO
GC
GE
GE
GE
GP
GT
HI
HL
IC
IF
IN
IN
IN
IP
IT
IT
IT
IT
IT
LA
LD
LE
LI
LO
LO

Error	Addr	Code	Seq	Source statement		
	7308	0D 0A	6867	DB	CR,LF	:
			6868			:
	730A	18	6869	CFMSG: DB	24	:
	730B	3F 34 30	6870	DB	'?40 SYSTEM BOOT FAILED'	:
	730E	20 20 53				
	7311	59 53 54				
	7314	45 4D 20				
	7317	42 4F 4F				
	731A	54 20 46				
	731D	41 49 4C				
	7320	45 44				
	7322	0D	6871	DB	CR	:
			6872			:
	7323	1D 0D 0A	6873	PUBMSG: DB	29,CR,LF	:
	7326	50 4F 57	6874	DB	'POWER UP BOOT IN PROGRESS'	:
	7329	45 52 20				
	732C	55 50 20				
	732F	42 4F 4F				
	7332	54 20 49				
	7335	4E 20 50				
	7338	52 4F 47				
	733B	52 45 53				
	733E	53				
	733F	0D 0A	6875	DB	CR,LF	:
			6876			:
	7341	10 0D 0A	6877	PWRMSG: DB	16,CR,LF	:
	7344	50 4F 57	6878	DB	'POWER RESTORED'	:
	7347	45 52 20				
	734A	52 45 53				
	734D	54 4F 52				
	7350	45 44				
			6879			
	7352	12	6880	SYNMSG: DB	18	:
	7353	3F 32 30	6881	DB	'?20 SYNTAX ERROR'	:
	7356	20 20 53				
	7359	59 4E 54				
	735C	41 58 20				
	735F	45 52 52				
	7362	4F 52				
	7364	0D	6882	DB	CR	:
			6883			:
	7365	12	6884	MEMMSG: DB	18	:
	7366	3F 34 30	6885	DB	'?40 MEMORY ERROR'	:
	7369	20 20 4D				
	736C	45 4D 4F				
	736F	52 59 20				
	7372	45 52 52				
	7375	4F 52				
	7377	0D	6886	DB	CR	:
			6887			:

Error	Addr	Code	Seq	Source statement		
7378	12		6888	COMMSG: DB	18	:
7379	3F 32 39		6889	DB	'?29 CPU TIME-OUT'	:
737C	20 20 43					
737F	50 55 20					
7382	54 49 4D					
7385	45 2D 4F					
7388	55 54					
738A	0D		6890	DB	CR	:
			6891			
738B			6892	CF_TL_MSG:		
738B	1B 3F 34		6893	DB	27,'?40 COMMAND FILE TOO LONG'	:
738E	30 20 20					
7391	43 4F 4D					
7394	4D 41 4E					
7397	44 20 46					
739A	49 4C 45					
739D	20 54 4F					
73A0	4F 20 4C					
73A3	4F 4E 47					
73A6	0D		6894	DB	CR	:
			6895			
73A7			6896	X_CMND_MSG:		
73A7	1E 3F 32		6897	DB	30,'?20 X COMMAND CHECKSUM ERROR'	:
73AA	30 20 20					
73AD	58 20 43					
73B0	4F 4D 4D					
73B3	41 4E 44					
73B6	20 43 48					
73B9	45 43 48					
73BC	53 55 4D					
73BF	20 45 52					
73C2	52 4F 52					
73C5	0D		6898	DB	CR	:
			6899			
73C6			6900	X_DATA_MSG:		
73C6	1B 3F 33		6901	DB	27,'?30 X DATA CHECKSUM ERROR'	:
73C9	30 20 20					
73CC	58 20 44					
73CF	41 54 41					
73D2	20 43 48					
73D5	45 43 48					
73D8	53 55 4D					
73DB	20 45 52					
73DE	52 4F 52					
73E1	0D		6902	DB	CR	:
			6903			
73E2	17		6904	CLKMSG: DB	23	:
73E3	3F 32 36		6905	DB	'?26 CPU CLOCK STOPPED'	:
73E6	20 20 43					
73E9	50 55 20					

ZZ
.M
RA

LO
LO
LO
LR
M
MA
MA
MA
ME
ME
MF
MI
MI
MI
MO
MS
MU
MY
NA
NE
NE
NE
NI
NO
NP
OC
ON
OU
PA
PA
PA
PC
PM
PN
PO
PR
PR
PU
PW
QT
RA
RC
RC
RC
RD
RE
RE
RE
RN
RT

Error	Addr	Code	Seq	Source statement
73EC	43 4C 4F			
73EF	43 4B 20			
73F2	53 54 4F			
73F5	50 50 45			
73F8	44			
73F9	0D	6906	DB	CR ;
		6907		
73FA		6908	CRD_ERR_MSG:	
73FA	19	6909	DB	25 ;
73FB	3F 34 30	6910	DB	'740 CRD TU-S8 NOT FOUND' ;
73FE	20 20 43			
7401	52 44 20			
7404	54 55 2D			
7407	35 38 20			
740A	4E 4F 54			
740D	20 46 4F			
7410	55 4E 44			
7413	0D	6911	DB	CR ;
		6912		
7414	03	6913	HLTMSG: DB	3 ;
7415	3F 30 30	6914	DB	'700' ;
		6915		
7418	0E 20 20	6916	PCMSG: DB	14,' PC='
741B	50 43 3D			
741E	30 30 30	6917	PCADR: DB	'00000000'
7421	30 30 30			
7424	30 30			
7426	0D	6918	DB	CR ;
		6919		
7427	50 57 52	6920	PUPF'L: DB	'PWRUP',0,'.CPU' ;
742A	55 50 00			
742D	2E 43 50			
7430	55			
7431	00 00 00	6921	DB	0,0,0,0,7 ;
7434	00 07			
		6922		
7436	01	6923	STRPAK: DB	DEPCOD
7437	03	6924	DB	03 ;GPR Space code
7438	000F	6925	DW	000FH ;Address of PC
743A	0000	6926	DW	0000 ;
743C	=0004	6927	SADRS: DS	4 ;
		6928		
		6929		
7440	0F 20 20	6930	UPCOUT: DB	15,' UPC='
7443	20 20 20			
7446	20 55 50			
7449	43 3D			
744B	0000 0000	6931	UPCBUF: DW	0,0
744F	0D	6932	DB	CR
		6933		

Error	Addr	Code	Seq	Source statement
			6934	
			6935	
7450	50 50 50		6936	CODTAB: DB 'PPPGMIUC'
7453	47 4D 49			
7456	55 43			
			6937	
7458	000F		6938	GPRCON: DW 000FH
745A	0000		6939	DW 0000H
745C	003F		6940	ICON: DW 003FH
745E	0000		6941	DW 0000H
7460	01FF		6942	MCON: DW 01FFH ;
7462	0000		6943	DW 0000H ;
7464	7FFF		6944	WCSCON: DW 07FFFH
7466	0000		6945	DW 0000H
7468	FFFF		6946	UCON: DW 0FFFFH
746A	0000		6947	DW 00000H
			6948	
746C	0000		6949	PNTR2: DW 0 ;
746E	=0004		6950	WRZBUF: DS 4 ;
			6951	
7472	=0004		6952	SHFBUF: DS 4 ;
			6953	
7476	00		6954	LPCNT1: DB 0
7477	00		6955	LPCNT2: DB 0
			6956	
7478	00		6957	SOURCE: DB 0 ;
7479	00		6958	DEST: DB 0 ;
			6959	
			6960	
747A	0A		6961	LSNWRN: DB 0AH
747B	00		6962	DB 0
747C	1B		6963	DB 01BH
			6964	
747D	0A		6965	WRNLSN: DB 0AH
747E	00		6966	DB 0
747F	1F		6967	DB 01FH
			6968	
7480	2A		6969	WRNWRN: DB 02AH
7481	00		6970	DB 0
7482	40		6971	DB 040H
			6972	
7483	15		6973	WRCRR: DB 015H ;
7484	FE		6974	DB 0FEH ;
7485	10		6975	DB 010H ;
			6976	
7486	15		6977	QTOWR: DB 015H ;
7487	80		6978	DB 080H ;
7488	A1		6979	DB 0A1H ;
			6980	
7489	15		6981	WRTQ: DB 015H ;

Error	Addr	Code	Seq	Source statement	
748A	C0		6982	DB	0C0H
748B	AA		6983	DB	0AAH
			6984		
748C	15		6985	CLRWR0: DB	015H
748D	80		6986	DB	080H
748E	2F		6987	DB	02FH
			6988		
748F	15		6989	COMWR0: DB	015H
7490	C0		6990	DB	0C0H
7491	A0		6991	DB	0A0H
			6992		
7492	15		6993	ASLWR0: DB	015H
7493	D0		6994	DB	0D0H
7494	23		6995	DB	023H
			6996		
7495	15		6997	RTLWR0: DB	015H
7496	C0		6998	DB	0C0H
7497	A3		6999	DB	0A3H
			7000		
7498	01 03		7001	SPPACK: DB	DEPCOD.3
749A	000E 0000		7002	SPADRS: DW	0EH.0
749E	=0004		7003	MEMADR: DS	4
74A2	=0004		7004	MEMSIZ: DS	4
			7005		
			7006		
74A6	00		7007	GRPCNT: DB	0
74A7	00		7008	BASADR: DB	0
74A8	00		7009	CONSIZ: DB	0
			7010		
74A9	0001 0000		7011	BASCON: DW	1.0
			7012		
			7013		
74AD	01		7014	CONBUF: DB	1
74AE	FF		7015	DB	0FFH
74AF	01		7016	DB	1
74B0	00		7017	DB	0
			7018		
74B1	01		7019	DB	1
74B2	93		7020	DB	93H
74B3	04		7021	DB	4
74B4	0000 0000		7022	MEMCOD: DW	0.0
			7023		
74B8	15		7024	DB	21
74B9	50		7025	DB	050H
74BA	01		7026	DB	1
74BB	00 03 05		7027	DB	0.3H.05H.06H.07H
74BE	06 07				
74C0	09 0B 0C		7028	DB	09H.0BH.0CH.0DH
74C3	0D				
74C4	0E 0F 13		7029	DB	0EH.0FH.013H.01FH

Error	Addr	Code	Seq	Source statement
74C7	1F			
74C8	34 3B 3F		7030	DB 034H,03BH,03FH,04BH ;Data
74CB	4B			
74CC	66 A0 F0		7031	DB 066H,0A0H,0F0H,0FFH ;Data
74CF	FF			
			7032	
74D0	08		7033	DB 8 ;Number of entries in second group
74D1	65		7034	DB 065H ;Base address for second group
74D2	02		7035	DB 2 ;Bytes per entry
74D3	01FF		7036	DW 01FFH ;Data
74D5	0FFF		7037	DW 0FFFH ;Data
74D7	2001		7038	DW 02001H ;Data
74D9	3000		7039	DW 03000H ;Data
74DB	7F80		7040	DW 07F80H ;Data
74DD	C000		7041	DW 0C000H ;Data
74DF	FFE0		7042	DW 0FFE0H ;Data
74E1	FFFF		7043	DW 0FFFFH ;Data
			7044	
			7045	
74E3	0B		7046	DB 11 ;Entries in next group
74E4	6D		7047	DB 06DH ;Base address for this group
74E5	04		7048	DB 4 ;Bytes per entry
74E6	0000 001F		7049	DW 0,01FH ;Data
74EA	0001 0020		7050	DW 1,020H ;Data
74EE	7000 00F2		7051	DW 07000H,0F2H ;Data
74F2	0000 0300		7052	DW 0,0300H ;Data
74F6	0000 041F		7053	DW 0,041FH ;Data
74FA	0000 0700		7054	DW 0,0700H ;Data
74FE	FE00 3FFF		7055	DW 0FE00H,03FFFH ;Data
7502	0000 7F80		7056	DW 0,07F80H ;Data
7506	FE00 7FFF		7057	DW 0FE00H,07FFFH ;Data
750A	FF0E 7FFF		7058	DW 0FF0EH,07FFFH ;Data
750E	0001 BF80		7059	DW 1,0BF80H ;Data
			7060	
7512	04		7061	DB 4 ;Entries in next group
7513	19		7062	DB 019H ;Base address for this group
7514	04		7063	DB 04 ;Bytes per entry
7515	F000 FFFF		7064	DW 0F000H,0FFFFH ;Data
7519	FF00 FFFF		7065	DW 0FF00H,0FFFFH ;Data
751D	FFFC FFFF		7066	DW 0FFFC H,0FFFFH ;Data
7521	FFFF FFFF		7067	DW 0FFFFH,0FFFFH ;Data
			7068	
7525	0A		7069	DB 10 ;Entries in group
7526	D0		7070	DB 0D0H ;Base address
7527	01		7071	DB 1 ;Bytes per entry
7528	14		7072	DB 014H ;Data
7529	18		7073	DB 018H ;Data
752A	1C		7074	DB 01CH ;Data
752B	24		7075	DB 024H ;Data
752C	28		7076	DB 028H ;Data

Error	Addr	Code	Seq	Source statement
752D	2C		7077	DB 02CH ;Data
752E	2D		7078	DB 02DH ;Data
752F	30		7079	DB 030H ;Data
7530	F8		7080	DB 0F8H ;Data
7531	FC		7081	DB 0FCH ;Data
			7082	
7532	08		7083	DB 8 ;Entries in group
7533	DA		7084	DB 0DAH ;Base address
7534	03		7085	DB 3 ;Bytes per entry
7535	2D20		7086	DW 02D20H ;Data
7537	00		7087	DB 0 ;Data
7538	0000		7088	DW 0 ;Data
753A	14		7089	DB 014H ;Data
753B	0000		7090	DW 0 ;Data
753D	15		7091	DB 015H ;Data
753E	0000		7092	DW 0 ;Data
7540	16		7093	DB 016H ;Data
7541	0000		7094	DW 0 ;Data
7543	17		7095	DB 017H ;Data
7544	0000		7096	DW 0 ;Data
7546	18		7097	DB 018H ;Data
7547	0000		7098	DW 0 ;Data
7549	1E		7099	DB 01EH ;Data
754A	0A10		7100	DW 0A10H ;Data
754C	04		7101	DB 4 ;Data
			7102	
754D	04		7103	DB 4 ;Entries in group
754E	E2		7104	DB 0E2H ;Base address
754F	04		7105	DB 4 ;Bytes per entry
7550	00E0 0C00		7106	DW 0E0H,0C00H ;Data
7554	0000 0FFF		7107	DW 0,0FFFH ;Data
7558	FF00 B020		7108	DW 0FF00H,0B020H ;Data
755C	FF10 FFFF		7109	DW 0FF10H,0FFFFH ;Data
			7110	
7560	20		7111	DB 32 ;Entries in group
7561	20		7112	DB 020H ;Base address
7562	05		7113	DB 05 ;Special code
			7114	
7563	00		7115	DB 0 ;Terminating 0
			7116	
			7117	
7564	00		7118	WEFLAG: DB 0 ;
7565	0000		7119	CFATAL: DW 0 ;
7567	0000		7120	WFATAL: DW 0 ;
7569	0000		7121	WERROR: DW 0 ;
			7122	;
			7123	;.....
			7124	
			7125	
			7126	;.....

Error	Addr	Code	Seq	Source statement
-------	------	------	-----	------------------

			7127	;
			7128	;The Indirect Command File buffer starts here and is 2K (2048) bytes
			7129	;long. Do not use this area for anything else.
			7130	;
	=7600		7131	ORG 07600H ;
			7132	
7600	=0800		7133	INDBUF: DS 2048 ;
			7134	
			7135	
	=7E00		7136	ORG 7E00H ;
			7137	
7E00	00		7138	REMRSR: DB 0 ;
7E01	00		7139	REMRDB: DB 0 ;
			7140	
7E02	01		7141	REMXSR: DB 1 ;
7E03	00		7142	REMXDB: DB 0 ;
			7143	
7E04	00		7144	TLKFLG: DB 0 ;
			7145	
7E05	00		7146	DISCRD: DB 0 ;
			7147	
7E06	00		7148	TRNFLG: DB 0 ;
			7149	
7E07	00		7150	MSGTBS: DB 0 ;
			7151	
7E08	0000		7152	TIMSAV: DW 0 ;
7E0A	0000		7153	TIMER1: DW 0 ;
7E0C	00		7154	TIMER2: DB 0 ;
7E0D	00		7155	TIKTOK: DB 0 ;
			7156	
7E0E	00		7157	LOGCON: DB 0 ;
7E0F	00		7158	LOGCNT: DB 0 ;
			7159	
7E10	00		7160	RDCON: DB 0 ;
7E11	00		7161	RDCON1: DB 0 ;
			62	
7E12	00		7163	RDTEMP: DB 0 ;
			7164	
7E13	01		7165	TALKSR: DB 1 ;
7E14	00		7166	TALKDB: DB 0 ;
			7167	
	=7E67		7168	ORG 7E67H ;
			7169	
7E67			7170	OVERRIDE LCS:
7E67	00		7171	DB 0 ;
			7172	
			7173	
			7174	;*****
			7175	;
			7176	; REVISION HISTORY

Error Addr Code

Seq Source statement

```
7177 :  
7178 : This is a Most Recent to Least Recent list of changes to the Console  
7179 : Ram software. The format is:  
7180 :  
7181 : VERSION NUMBER where the change was first implemented  
7182 : DATE OF CHANGE  
7183 : AUTHOR OF CHANGE  
7184 : NUMBER OF CHANGE  
7185 : LOCATION OF CHANGE given as a label +/- a number of lines.  
7186 : DESCRIPTION OF CHANGE  
7187 :  
7188 :  
7189 :  
7190 :  
7191 : 01-JAN-1984 Vijay Lathia  
7192 :  
7193 : #1  
7194 :  
7195 : TMRPR0:(+1) and TMRPR0:(+8)  
7196 :  
7197 : Code added to block and unblock 9513 before accessing it  
7198 :  
7199 : #2  
7200 :  
7201 : TPSUB:(+3)  
7202 :  
7203 : Code added to deposit 05F to ITCSR to init 9513 correctly  
7204 :  
7205 :  
7206 :  
7207 : 91.00 15-MAR-1982 John Middleton  
7208 :  
7209 : #1  
7210 :  
7211 : ITWCS2:(-8) and ITRUN1:(+4)  
7212 :  
7213 : INR M changed to INR A.  
7214 :  
7215 : #2  
7216 :  
7217 : TMRSR4:(+1)  
7218 :  
7219 : Put the LARM12 value in A and then do an OUT to the ITCSR.  
7220 :  
7221 :  
7222 : 90.00 03-MAR-1982 John Middleton  
7223 :  
7224 : #1  
7225 :  
7226 : TTRTSA:(-1) to TTRTSA:
```

Error Addr	Code	Seq	Source statement
7227	:		
7228	:		Must keep calling Get Silo until the LRMASK gets altered.
7229	:		
7230	:		
7231	:	89.01	02-MAR-1982 John Middleton
7232	:		
7233	:		#1
7234	:		
7235	:		TURDB:(+7) to TURDB0:
7236	:		
7237	:		Only clear the errors by sending a 15H to the TUCR if error
7238	:		flags are actually on.
7239	:		
7240	:		
7241	:		
7242	:	89.00	27-OCT-1981 John Middleton
7243	:		
7244	:		#1
7245	:		
7246	:		HALT:(+6) to HALT:(+7)
7247	:		
7248	:		Print Halt PC for H(alt) command.
7249	:		
7250	:		
7251	:	88.04	9-NOV-1981 John Middleton
7252	:		
7253	:		#1
7254	:		
7255	:		PMHALT:(+7)
7256	:		
7257	:		(check for characters in XMTDB and INTDB and, if there are any,
7258	:		print them before returning to Console Mode.
7259	:		
7260	:		
7261	:	88.03	28-OCT-1981 John Middleton
7262	:		
7263	:		#1
7264	:		
7265	:		PARIT1:(+3)
7266	:		
7267	:		Must restore PSW before leaving
7268	:		
7269	:		
7270	:	88.02	26-OCT-1981 John Middleton
7271	:		
7272	:		#1
7273	:		
7274	:		TMRSR1:(-3)
7275	:		
7276	:		Call XFRSE0 instead of XFRSET. The decision made at XFRSET has

Error	Addr	Code	Seq	Source statement
			7277	;
			7278	;
			7279	;
			7280	;
			7281	;
			7282	;
			7283	;
			7284	;
			7285	;
			7286	;
			7287	;
			7288	;
			7289	;
			7290	;
			7291	;
			7292	;
			7293	;
			7294	;
			7295	;
			7296	;
			7297	;
			7298	;
			7299	;
			7300	;
			7301	;
			7302	;
			7303	;
			7304	;
			7305	;
			7306	;
			7307	;
			7308	;
			7309	;
			7310	;
			7311	;
			7312	;
			7313	;
			7314	;
			7315	;
			7316	;
			7317	;
			7318	;
			7319	;
			7320	;
			7321	;
			7322	;
			7323	;
			7324	;
			7325	;
			7326	;

already been made by the TMRSRV code. Changing the entry point saves 10us on the worst case TMRSRV time.

88.01 26-OCT-1981 John Middleton

#1

TMRSR2:(+4)
TMRSR4:(+0) to TMRSR4:(+6)

Mask RST 5.5 if the check of the Timer RUN bit while in TMRSRV reveals that the timer has been turned off.

88.00 25-OCT-1981 John Middleton

#1

TMRSR2:(+4)

If the check of the Timer RUN bit while in TMRSRV finds that the timer has been turned off:

1. Load and arm counters 1 and 2. This should make the counter and alarm values unequal.
2. Drive SOD low, then high. This should drive the 9513 interrupt to the 8085 low.

87.02 -OCT-1981 John Middleton

#1

ENTER:(+6) to ENTER:(+15)

When entering Program Mode, make sure the 2ms timer (counter #3 of the 9513) is off and the overflow has been cleared. This fixes the error where the "Power Restored" message was being printed as the result of executing a Start command.

87.01 -OCT-1981 John Middleton

#1

ACFAI1:(+2)

Conditional Jump was wrong polarity. Console would hang if Standby was entered while in Console Mode.

* * Symbol Table * *

A	Reserved	AACTIV	0040	AAFLAG	0042	ABUFFR	71FE	ACFAI0	5DFD
ACFAI1	5E05	ACFAI2	5E0C	ACFAI3	5E33	ACFAIL	5DE5	ACHECK	4E2D
ACHK1	4E36	ACLO	4120	ACVECT	4C25	ADDAD1	6251	ADDADR	624E
ADRES1	7114	ADRES2	7145	ADRES3	716A	ADRSPC	71B5	ADRTS1	4FBA
ADRTST	4FB2	AFLAG	0002	AFTER	557C	AFTER1	5597	ALARM	7090
ALARM1	0007	ALLOWP	0003	ALRMM3	7094	ALTCHK	4B2E	ALTFLG	4181
ALTMOD	001B	ANDAD1	56AA	ANDADR	56A5	APTFLG	0004	APTLOD	40D6
ARM12	0023	ARM45	0038	ASLWR0	7492	ASMASK	000F	ASTRSK	4FA0
ATFLAG	0080	ATFLOW	52DF	ATSIGN	4F9B	B	Reserved	BACKSL	005C
BASADR	74A7	BASCON	74A9	BASE	007B	BEFORE	555B	BEGIN	0000
BLKFLG	7098	BLKNUM	4174	BLOCK	4097	BMPPNT	471D	BOOT	576D
BOOTEN	0007	BOOTSW	0001	BPHALT	7222	BPSUB	57D5	BREAK	000D
BRKCNT	7221	BRKFLG	7220	BSPREP	57E4	BUFADR	40CA	BYTCNT	4095
BYTE	4E92	BYTE00	417A	BYTE01	417B	BYTES	416E	BYTSUM	4082
C	Reserved	CCFLAG	40CF	CFAIL	6062	CFATAL	7565	CFERR	0002
CFLOW	535B	CFMSG	730A	CF_ERR	5D16	CF_TL	738B	CGIMOD	0001
CGFIVE	000D	CGFOUR	000C	CHANGE	5DD7	CHECK	652D	CHKSUM	00DE
CHRBUF	7001	CHRCNT	7000	CHRSIL	4B80	CLDMSG	72EB	CLDST0	6027
CLDST1	602F	CLDSTR	6024	CLDSTX	6020	CLKERR	5D70	CLKFLG	4196
CLKMSG	73E2	CLITLK	4A46	CLOCK	0006	CLRACK	00A9	CLRACL	0033
CLRATN	00AB	CLRBLK	003A	CLRBSY	002E	CLRCLD	6AE7	CLRCLK	0020
CLRCCK	0024	CLRC3R	0038	CLRDCL	0031	CLRHLT	00AF	CLRLIT	003C
CLRMCI	0029	CLRMCK	002B	CLRPAG	003F	CLRPAR	6994	CLRPFI	00AD
CLRPME	6AF8	CLRSS	0022	CLRTIM	00A5	CLRTMR	002C	CLRUPC	00A8
CLRUPK	0026	CLRWCK	0036	CLRWRO	748C	CLRWRM	6AF1	CMDVAL	4182
CMPADR	4162	CMPAR1	7102	CMPAR2	7139	CMPAR3	715E	CMPCH1	6244
CMPCHK	6242	CMPTB1	4183	CMPTB2	4185	CNSOLA	4C42	CNSOLB	4C5E
CNSOLX	4C7B	CNTADR	40C8	CNTINO	58E2	CNTIN1	586B	CNTIN2	5877
CNTRLR	42AC	CNTRLR1	42BD	CNTLU	42CE	CNTRLC	0003	CNTRLO	000F
CNTRLP	0010	CNTRLQ	0011	CNTRLR	0012	CNTRLS	0013	CNTRLU	0015
CNTRLX	0018	CNVER1	4E0A	CNVERT	4DF4	CODFIL	7247	CODFL1	4C2E
CODFLG	408B	CODTAB	7450	COLD	408A	COLD_B	5F99	COLD_B	5FD2
COLD_B	5FE8	COLD_H	601A	COLON	003A	COMERR	5D5E	COMFIN	568A
COMM1	4E9E	COMMON	4E8C	COMMSG	7378	COMM_C	0005	COMWRO	748F
CONSMS	1000	CONBUF	74AD	CONHL1	58CA	CONHL2	58A3	CONHL3	58C1
CONHLT	587A	CONMOD	71FD	CONMSG	41DF	CONPAK	71FC	CONSLZ	74A8
CONSOL	41CE	CONT	5840	CONTO	584B	CONTLR	41BE	CONTLU	41BA
COPY	00F9	COPY1	00FB	COPYB0	419C	COPYB1	419D	COUNT	4187
COUNT1	7101	COUNT2	7138	COUNT3	715D	CPATTN	0083	CPCH0	62BD
CPCH1	62D0	CPCHK1	629D	CPCHK2	6375	CPCHK3	639A	CPCHK4	63BE
CPCHK5	64A9	CPCHK6	64CB	CPFLAG	40F3	CPSER1	6560	CPSER2	656D
CPSERV	6558	CPUACK	0082	CR	000D	CRAFIL	7279	CRCMD	5D03
CRD_CO	0001	CRD_ER	5D78	CRD_ER	73FA	CRD_FL	40FD	CRD_RD	4B3C
CRD_VE	0398	CRFLOW	4FD6	CRMFIL	728B	CSR07	0085	CSRIS	0086
CSR23	0087	CSRBUF	41A4	CTLDIS	40FE	CVECTR	4C28	D	Reserved
DACTIV	0080	DARM12	0083	DATCHK	6233	DATLEN	71B4	DBUFFR	7202
DCHK1	4E28	DCLO	0002	DDECOD	4FDE	DECOD1	4CFB	DECODE	4CFA
DECODX	52AE	DEFDRV	4150	DEFFIL	7255	DELAY1	0457	DELETE	007F
DELFLG	40CE	DELMSK	00FE	DEPADR	71D2	DEPCOD	0001	DEPDAT	71D6
DEPIO	50B6	DEPLS1	54C9	DEPLS2	54CC	DEPPAK	71D0	DEPHRO	5543

• • Symbol Table • •

DEST	7479	DEVFLG	7212	DFAULT	52D5	DFLAG	0004	DIFUNI	4544
DIR1	5026	DIRADR	417C	DIRBUF	4300	DIRECT	4FEF	DIRFLG	409C
DIRSEG	42F6	DIRUNI	4195	DIRVAL	4194	DIRVEC	411A	DISCNT	40DB
DISCRD	7E05	DISMEM	00A7	DISPAR	00A1	DIV50	4857	DIVSR1	F9C0
DIVSR2	FFD8	DIVSR3	FFF0	DLMASK	00F0	DLMCHK	4E20	DNPRES	5339
DNPREP	5329	DOT	4E3B	DOTBUF	410D	DPOS1	5065	DPOS2	5068
DPOSIT	5034	DRVFIL	41D8	DUMMY1	0052	E	Reserved	EDADDR	71B8
EDCSR	53E5	EDCSR1	53FD	EDDATA	71BC	EDLS	549E	EDMOD	71B7
EDPACK	71B6	EDPAR	540E	EDQ	54FA	EDWR	5517	ENBMEM	00A6
ENBPAR	00A0	ENDPAK	40AA	ENDSE1	4787	ENDSEG	476D	ENTER	625F
ENTYPE	4178	ERROR	5D80	ERROR1	5DA2	EXAMI1	511F	EXAMI2	5122
EXAMIN	50F2	EXAMQ	5511	EXE1	5663	EXECUT	5649	EXMCOD	0000
EXMIO	51F1	EXMLS	54D1	EXMLS1	54EC	EXMPAR	5435	EXMWR	554F
EXTBUF	410E	EXTCNT	7211	EXTFLW	4DDA	EXTPTR	720E	EXTRAS	42FC
EXTYPE	523A	FASTWR	5F4D	FASUPC	0016	FASWR1	5F65	FASWR2	5F78
FIXCP1	46B3	FIXCPU	46A6	FIXER1	466B	FIXER2	4685	FIXERR	4663
FIXUP	5A0D	FLAGS	7213	FLAGS1	7214	FLAGS2	7215	FLAGS3	7216
FLIP	543D	FLIP1	547F	FNBUF	4107	FNCNT	7210	FNDFLG	4164
FNFLOW	4D8D	FNFLW1	4DC0	FNFLW2	4DD3	FNFMMSG	41EF	FNPTR	720C
FOOFIL	7267	FOUR	5CCA	FOURDN	5343	FOURUP	5317	FU_VEC*	004F
GCVECT	4129	GETS12	47E0	GETCHK	4CC6	GETCS	5B67	GETDA1	5CCF
GETDAT	5CAC	GETEND	492F	GETLI0	4A58	GETLI1	4A7A	GETLI2	4AA0
GETLI3	4AAD	GETLI4	4A91	GETLI5	4ABC	GETLI6	4AD3	GETLI7	4AF5
GETLI8	4AF6	GETLIA	4A74	GETLIN	4A4D	GETNXT	4FC5	GETON1	47A8
GETON2	47AE	GETON3	47D4	GETON4	47D7	GETONE	478C	GETTWO	484A
GETUPC	59B6	GFLOW	5355	GLVECT	4123	GOSV	4103	GO_STA*	4C00
GPR	4E76	GPRCON	7458	GRPCNT	74A6	GSVECT	4145	GS_VEC*	0046
GT512A	47E5	H	Reserved	HALT	58E8	HALTPC	71DC	HEXASC	5DBD
HIGHST	42FA	HLBUFF	4088	HLTBUF	71DA	HLTCOD	71DB	HLTFLG	0002
HLTMSG	7414	HLTPSL	71E0	HMPEND	707F	HOLD1	0019	HOLD4	001C
ICCS	707C	ICHEC1	53AC	ICHECK	53A4	ICON	745C	IDLE	4C32
IFLAG	7223	IFLOW	5367	IMPXFR	7099	INCCNT	417F	INDADR	721C
INDBUF	7600	INDCNT	721B	INDFLG	409D	INDRC1	5A23	INDRCT	5A1B
INDTS1	4CA7	INDTS2	4CB6	INDTST	4C94	INISUB	56F8	INIT	56E6
INITH	0035	INITL	0034	INITPK	71FB	INIVEC	0040	INTACK	6575
INTCSR	7074	INTDB	7075	IOARG1	50BF	IOARG2	51F7	IPR	7071
IPRCHK	66B6	IREG	4E80	ISR	7070	ITCSR	001D	ITDB	001C
ITEST	537D	ITEST1	5388	ITEST2	539C	ITICR	68B2	ITICR0	68C1
ITICR1	68CE	ITICR2	68FD	ITICR3	68ED	ITNICO	6947	ITNIC1	694D
ITNICR	691B	ITRCR1	689B	ITRUN	679E	ITRUN0	67F4	ITRUN1	67FF
ITRUN2	67F6	ITSGL	707E	ITTEMP	709E	ITWCS0	66E6	ITWCS1	66E8
ITWCS2	671F	ITWCS3	6747	ITWCS4	6752	ITWCS5	6768	ITWCS6	6789
ITWCS7	6793	ITWCSR	66C6	L	Reserved	LA1245	007B	LARM12	0063
LARM2	0062	LARM3	0064	LARM45	0078	LAST	6590	LCMASK	00DF
LDCSR	49A4	LDCSR1	49A7	LDUPC	49D1	LDUPC1	49D4	LDXPRP	4500
LENGTH	4166	LF	000A	LIMIT	41B4	LIMSP	41B5	LIST	46D9
LIST1	46DC	LIST2	4710	LIST3	4713	LIST4	472F	LIST5	4737
LIST6	4759	LIST7	46EC	LIST8	46E3	LNGSAV	71E8	LNGSUM	71EC
LOAD	58FC	LOAD1	45C3	LOAD12	0043	LOAD1X	45F9	LOAD2X	4625
LOAD3X	463A	LOAD45	0058	LOAD4X	4656	LOADA	45A1	LOADB	45B6

• • Symbol Table • •

LOADED	7082	LOADX3	454E	LOADZ	58FF	LOADADR	40A3	LODCNT	40A7
LODDST	4165	LODFLG	409B	LODSU1	5925	LODSUB	590A	LODVEC	4117
LOGCNT	7E0F	LOGCON	7E0E	LONG	4E9C	LPCNT1	7476	LPCNT2	7477
LRMASK	40BE	LSHIFT	4A00	LSNWRN	747A	LSTBUF	41C2	LSTPRP	46CB
M	Reserved	MAKCN1	5EB8	MAKCN2	5ECC	MAKCN3	5EDC	MAKCN4	5F16
MAKCN5	5F31	MAKCN6	5F39	MAKCN8	5EE5	MAKCON	5E9E	MAKEWR	5604
MAKMOD	5373	MAKSPC	5293	MARKSP	5DB1	MASTER	0017	MATCH	4D07
MAXCNT	0050	MCHECK	53BA	MCON	7460	MEMADR	749E	MEMCOD	74B4
MEMDEP	508F	MEMERR	5D46	MEMEXM	51C1	MEMLD1	5971	MEMLOD	5957
MEMMSG	7365	MEMSIZ	74A2	MEM_CO*	0004	MERGE	52FB	MFLOW	536D
MFLOW1	542A	MICCON	580C	MICDEP	50A5	MICEXM	51E0	MICFLG	721F
MICMON	4154	MICPRO	4207	MICRO	4E85	MICST1	598F	MICST2	599C
MICST3	599F	MICST4	59A6	MICSTP	5976	MINCNT	708C	MINUS	4F96
MIPFLG	40D9	MISC2	0001	MKSPC1	5296	MMSTRT	4C00	MODADR	4099
MODIFR	4090	MODVEC	5555	MREG	4E7B	MRKBUF	7219	MSGADR	40B7
MSGBUF	40F0	MSGCNT	408E	MSGTBS	7E07	MSKCPY	40BF	MSTRCD	0A0C
MUL2	4823	MULSH1	46C0	MULSHF	46BE	MYCNT	4179	MYLDER	4266
MYLOAD	420C	MYPC	40B9	NACKCM*	0007	NACKDA*	0008	NACK_X*	5D4E
NACK_X*	5D56	NAMES	4302	NBUFFR	7208	NEG1	7196	NEG2	719A
NEG200	71A6	NEG3	719E	NEG4	71A2	NEWAS	71B3	NEWCN0	6874
NEWCN1	6878	NEWCN2	687A	NEWCNT	684C	NEWDL	71B2	NEXTA1	4EB9
NEXTA2	4EDC	NEXTA3	4EDF	NEXTAD	4EA4	NICFLG	7083	NICR	7084
NICRM3	7088	NICSUB	6953	NOCHK	40FF	NOCON	58DB	NOPBUF	41A7
NOTBYT	5281	NOTL	4F65	NOTPVG	5289	NOTYPE	40D4	NPREP	5089
NPREP1	5145	NSADBF	7206	NTEST	56BE	NTEST1	56D2	NXTSEG	42F8
OCOPY	40BC	OFLAG	40BB	OKSV	0005	OKV15	0007	OLDRUN	707D
ONE	5CBB	ONEDN	534F	ONEUP	5323	OPCODE	408F	OUTADR	7227
OUTBUF	7224	OUTDAT	7230	OUTLET	7225	OVERRI*	7E67	PACKS	4172
PAKCNT	40A8	PARERR	411D	PARFLG	4197	PARITY	5E56	PARMSK	007F
PARSE	4D2B	PARSE1	4D33	PARSE2	4D50	PARSE3	4D53	PARSE4	4D71
PARSES	4D83	PARTMP	4198	PARVEC	4C22	PCADR	741E	PCMSG	7418
PCOUNT	70FC	PFFLAG	40B5	PHYS	4E6C	PHYVIR	52E8	PLUS	4F91
PMBOOT	6ADA	PMHAL0	6647	PMHALT	6625	PMTALK	6441	PMTLK0	6458
PNTR1	41A2	PNTR2	746C	POINTR	4176	POS1	7186	POS2	718A
POS200	71 2	POS4	718E	PRGCNT	4F79	PRI1	66A5	PRI2	66AA
PRI3	66AF	PRI4	66B4	PRIADR	40C1	PRICNT	40C3	PROMPT	4203
PRPSUB	4CCF	PSLADR	0000	PSW	Reserved	PUBCHK	5F8B	PUBFLG	40B6
PUBMSG	7323	PUPFIL	7427	PUVEC	4151	PVECTR	4C2B	PWRFIL	7239
PWRMSG	7341	PWRVE1	004C	PWRVEC	001B	PXXX	4F4B	QPEND	40D8
QTOWR	7486	QUAL	4115	QUAL1	7218	QUO	4160	RAD50	482B
RAD50A	483A	RADCVT	4895	RAMADR	71AA	RAMCNT	71AE	RAMVAL	4080
RCVADR	71C2	RCVBUF	71C0	RCVCSR	7072	RCVDAT	71C6	RCVDB	7073
RCVDON	0002	RCVMOD	71C1	RCVMS1	518E	RCVMS2	519B	RCVMS3	51A3
RCVMS4	51B0	RCVMS5	518C	RCVMSG	5189	RCVMSX	5186	RCVTTR	637B
RDCON	7E10	RDCON1	7E11	RDTEMP	7E12	RDWRZ	55A4	RDWRZ1	55AC
READ	0080	READJ2	0004	REENTR	52BA	REMAIN	415E	REMOTE	0006
REMRDB	7E01	REMRSR	7E00	REMXDB	7E03	REMXSR	7E02	REPEAT	4D0E
RESADR	71E4	RETRY	408C	RETRY1	40CC	RET_VE*	414D	RFLAG	7217
RN	4F6D	RSHF	5627	RSHF1	562A	RSVEC	0049	RSVECT	413D
RTEST	56B4	RTEST1	56BC	RTLWR0	7495	RUBOU1	4B22	RUBOUT	4B0E

• • Symbol Table • •

RUNFLG	40D3	SADRS	743C	SAVADR	41B8	SAVCHR	40C0	SAVCNT	4199
SAVE	419E	SAVE12	00A3	SAVE45	00B8	SAVEB0	419A	SAVEB1	419B
SAVESP	0104	SAVMSK	40DA	SAVSP1	0108	SAVUPC	41AE	SBUFFR	4111
SCNADR	71CC	SCNPAK	71CA	SCVECO	0043	SCVECT	4138	SECURE	0003
SEGNUM	417E	SEMI	4FCE	SENDQ	6549	SENDS	6531	SEQNUM	4093
SETACK	00A8	SETACL	0032	SETAPT	4288	SETATN	00AA	SETBLK	003B
SETBSY	002F	SETCLK	0021	SETCSK	0025	SETCSR	0039	SETDCL	0030
SETHLT	00AE	SETLIT	003D	SETMCI	0028	SETMCK	002A	SETPAG	003E
SETPAR	698F	SETPFI	00AC	SETPME	6AF9	SETSS	0023	SETTIM	00A4
SETTHR	002D	SETUPC	00A9	SETUPK	0027	SETVAL	4296	SETMCK	0037
SFLAG	40CD	SHARE	4F8B	SHAREX	4FA2	SHFBUF	7472	SHIFT	4E0D
SHIFT1	4E10	SIBIAS	40D1	SILFLG	40D0	SKIP40	5DE2	SKP11	5256
SKP319	48B5	SNP320	48AB	SKP321	48A6	SKP523	48B7	SLASH	4E56
SLOW C	40F4	SLVEC1	4133	SLVECT	412E	SLMCLK	0006	SMLALM	7081
SMLNIC	7080	SNDADR	40A0	SNDCNT	40A2	SNDCTL	653E	SNDDAT	5CD9
SNDDEL	4B04	SNDFLG	40F2	SNDMS0	5150	SNDMS1	5157	SNDMS2	515F
SNDMS3	516C	SNDMS4	5174	SNDMS5	5181	SNDMSG	514E	SNDMSX	514B
SNDPAK	40BD	SNDPR1	6214	SNDPRP	6204	SNDRCV	6222	SNDRCX	622B
SNDTA	646B	SNDTL	6494	SNDTL0	649C	SNDTL1	64A4	SNDTL2	64A6
SNDTL3	64A7	SNDTR	6475	SNDTRL	6483	SNDTRL	641E	SNDTT0	63E0
SNDTT1	63ED	SNDTT2	63FD	SNDTTX	63C4	SNTRL1	642C	SOBIAS	40D2
SOFDOW	6AD7	SOURCE	7478	SP	Reserved	SPACE	0020	SPADRS	749A
SPBUFF	40B6	SPCF1	4E4F	SPCFLG	0001	SPCFLW	4E4C	SPEND	40D7
SPPACK	749B	SPSTRP	4D23	SRCFIL	4BC0	STACK	4080	STAL10	5760
STALLS	5763	START	4100	START1	5815	START2	582D	STEP	59D6
STEP1	00F1	STEP2	00F2	STPA	59EA	STEPB	59F8	STEPB1	59FD
STPC	59F5	STKFLW	4F7E	STPFLG	721E	STRBLK	42FE	STRPAK	7436
STRTA1	4F06	STRTA2	4F2E	STRTA3	4F45	STRTAD	4EF1	SUBDIV	4875
SUBDV1	487B	SUBDV2	488B	SUCCE	4116	SUMREG	0020	SUPRES	526C
SUPRS4	5284	SUPRS6	527C	SWAP	41AA	SWITCH	4092	SWPOS	4081
SYNERR	5D1E	SYMSG	7352	SYN_CO	0003	S_COMM	57F2	TAB	0009
TABLE	70A2	TALKDB	7E14	TALKSR	7E13	TALK_V	080F	TAPERR	5D2D
TEMP	40B4	TEMP1	416A	TEST	5A3E	THREE	5CC5	TIKTOK	7E0D
TIMCNT	7051	TIMER1	7E0A	TIMER2	7E0C	TIMSAV	7E08	TLKFLG	7E04
TMPBUF	416B	TMRFIL	7052	TMRPR0	69BF	TMRPR1	69D7	TMRPR2	6999
TMRSR0	6A28	TMRSR1	6A4B	TMRSR2	6A55	TMRSR3	6A6A	TMRSR4	6A6E
TMRSRV	6A18	TMRVEC	4C2F	TOCNT	4083	TOCNT1	4085	TODBFR	709A
TODPRP	6AB5	TODR	6A7D	TODW	6A8D	TODW1	6A93	TPIN11	0058
TPINIT	0055	TPSUB	69DE	TPSUB0	69F2	TPSUB1	69F4	TRCR	0043
TRDB	0040	TRKUP1	41B2	TRKUPC	41B0	TRMODE	0042	TRNFLG	7E06
TRSR	0041	TRYCHK	4924	TTCR	0048	TTRDB	0048	TTRMODE	004A
TTRCSR	6595	TTRDB	6656	TTRDB1	6679	TTRID	6606	TTRIE	65C2
TTRRE1	6514	TTRREM	64F2	TTRTS0	6328	TTRTS1	632A	TTRTS2	633D
TTRTS4	6355	TTRTS5	6372	TTRTSA	6319	TTRTSB	62FE	TTRTST	62ED
TTSR	0049	TTXCSR	658C	TTXDB	65A0	TTXID	65FB	TTXIE	65BB
TTXTST	63A0	TUCR	0047	TUDB	0044	TUERR	0086	TUERR1	42A4
TUERR2	0079	TUERR3	0081	TUINIT	0004	TUMODE	0046	TUPREP	0015
TURCSR	659B	TURCV	48C0	TURCV1	48E0	TURCV2	4900	TURCV3	48C3
TURDB	6683	TURDB0	6696	TURID	661A	TURIE	65F4	TURTST	64D1
TUSR	0045	TUWCS1	65E0	TUWCSR	65C9	TUXDB	65B0	TUXID	6611

• • Symbol Table • •

TUXTST	64AF	TU_SEN*	0095	TWO	5CC0	TWODN	5349	TWOUP	531D
T_VECT*	414D	UBINIT	6ACD	UCON	7468	UFLOW	5361	UNIT	4091
UNITX	4106	UPC14	0084	UPC14A	0000	UPCADR	0001	UPCBUF	744B
UPCHK	5481	UPCEX1	5688	UPCEXM	566B	UPCFLG	0080	UPCOUT	7440
UPCTST	529E	UPPREP	5305	USED	0080	USTART	583C	VERMSG	729D
VNORML	00A3	VRTUAL	4E71	VSHIFT	00A2	WAICMD	5AAA	WAITH	6971
WAITL	6980	WARM	41B7	WCS	4E8A	WCSB0	0008	WCSB1	0009
WCSB2	000A	WCSCON	7464	WCSDE1	50F0	WCSDEP	50C3	WCSEX1	521F
WCSEX2	521D	WCSEXM	51FE	WCSLOD	4A09	WCS_LO*	40D5	WEFLAG	7564
WERROR	7569	WFADR	71F2	WFAIL	61E3	WFAIL1	61F2	WFATAL	7567
WFDATA	71F6	WFMSG	72CF	WFPACK	71F0	WORD	4E97	WRBYT0	4A36
WRBYT1	4A3E	WRCRR	7483	WRDSUM	409E	WRITE	00CC	WRKMSK	40BD
WRMCNT	71FA	WRMSG	72AD	WRMST1	609A	WRMST2	6104	WRMST9	6122
WRMSTR	6074	WRNLSN	747D	WRNWRN	7480	WRR0T8	5633	WRR0T8X	5638
WRTNOP	4993	WRT0Q	7489	WRTWCS	5D05	WRTWR1	55E1	WRTWR2	55EE
WRTWRZ	55CA	WRZBUF	746E	XADRES	40C4	XCOUNT	40C6	XCSADR	7078
XDBADR	707A	XFER	5AD8	XFER1	5ADD	XFRADR	418A	XFRBUF	4192
XFRCHK	623C	XFRCNT	418E	XFRCOM	5CE7	XFRPAK	4188	XFRA1	5BDA
XFRRAM	5BC2	XFRSE0	682B	XFRSET	6823	XFRWCS	5C27	XMEMH	5B28
XMEMH1	5B35	XMEM1	5B1C	XMEM11	5B48	XMEM12	5B51	XMEM13	5B61
XMEML	5B3E	XMEM0	5B7B	XMEM01	5BA6	XMEM02	5B94	XMEM04	5B86
XMTCSR	7076	XMTDB	7077	XOFIN	5BB7	XRAM11	5BE0	XRAM12	5BF0
XRAM13	5BF7	XRAM14	5BFC	XRAM0	5C01	XRAM01	5C11	XRAM02	5C1E
XRAM03	5C22	XVEC	000B	XWCSI	5C38	XWCSI1	5C5B	XWCS0	5C64
XWCS01	5C6F	XWCS02	5C9B	XWCS03	5CA1	X_CMND*	73A7	X_DATA*	73C6
YBUSRD	00EC	ZBUF1	6AFC	ZBUF2	6AFD	ZERO	7182	ZEROS	70FD
ZROBUF	6AFA	ZROCHK	623F						

Symbol Value References (# = Definition, \$ = Write, <BLANK> = R#)

A	Reserved	503\$	508\$	608	702	704\$	709\$	717\$	719	738\$	741\$	774	776\$	799\$
		830\$	831	834	859	861	936	938\$	985	999\$	1009\$	1012	1031\$	1040
		1047\$	1103\$	1105\$	1116\$	1129\$	1131	1132\$	1134	1141	1143\$	1149	1177\$	1215
		1218\$	1226	1233\$	1234	1240	1245\$	1253\$	1269\$	1287\$	1292\$	1301	1302\$	1304
		1305\$	1307	1349\$	1353\$	1366	1370\$	1374\$	1378	1407\$	1427\$	1449\$	1453\$	1457\$
		1469\$	1482\$	1541	1570\$	1572	1589\$	1590\$	1591	1593\$	1601	1602	1638	1654\$
		1673\$	1676	1685	1688\$	1705\$	1713\$	1718	1720\$	1726	1734\$	1739\$	1747	1748\$
		1780\$	1785\$	1788	1789	1790	1791	1830	1913	1923	1936	1954	1964\$	1969\$
		1970\$	1974\$	1977\$	1982	2020	2023	2039\$	2053\$	2085	2118\$	2126\$	2135\$	2138\$
		2149\$	2152\$	2153	2164\$	2167\$	2168	2216	2220	2225\$	2227	2318\$	2320\$	2322\$
		2324\$	2326\$	2328\$	2330\$	2338\$	2340\$	2342\$	2353	2375\$	2385	2409\$	2436\$	2441\$
		2450\$	2457\$	2481	2500\$	2503\$	2506\$	2508	2522\$	2550\$	2554\$	2566\$	2570\$	2576\$
		2585	2641	2650	2651	2658	2679	2693	2728	2731\$	2741\$	2768	2779	2784\$
		2785	2800	2808\$	2810	2817	2833	2856	2859\$	2863\$	2906\$	2928	2948\$	3062\$
		3076\$	3078	3080\$	3088\$	3090	3147	3150	3157	3165	3172	3177	3208	3216
		3223	3239	3248	3257	3268	3283	3314\$	3317	3346\$	3353	3357\$	3364\$	3380\$
		3394	3400	3407	3408\$	3410	3418\$	3423\$	3434\$	3440\$	3442	3461	3491	3516
		3527\$	3534	3538	3570	3592\$	3614\$	3628\$	3644	3647	3655	3668	3672\$	3684
		3687\$	3709\$	3712	3741\$	3747\$	3748	3752	3764\$	3772\$	3781\$	3791	3804	3836
		3846	3863	3864	3866\$	3878\$	3881	3903	3905\$	3908\$	3924\$	3928	3957\$	3971\$
		3975	3991\$	3995\$	3996	4025\$	4027	4042\$	4055\$	4057\$	4059\$	4061\$	4067	4072
		4104	4112\$	4117	4174	4201\$	4205\$	4207	4225\$	4233\$	4235\$	4245\$	4248\$	4255\$
		4310	4317\$	4325	4329	4334	4339	4344	4349	4350\$	4351\$	4363	4367\$	4369
		4370\$	4373	4375\$	4384	4393\$	4396\$	4399\$	4412\$	4416	4419\$	4423	4430\$	4434\$
		4439\$	4444\$	4451	4453\$	4457\$	4462\$	4470	4473	4484	4485\$	4496	4508\$	4517\$
		4529\$	4544	4546\$	4550\$	4557\$	4568\$	4570\$	4573	4583\$	4586\$	4597\$	4599\$	4614
		4628\$	4640	4659\$	4662	4668\$	4670	4674\$	4677\$	4686	4690	4704	4705\$	4714\$
		4722	4723	4725	4729\$	4731	4757\$	4774\$	4786	4799	4802\$	4814\$	4820\$	4824\$
		4830\$	4835\$	4849	4854	4856\$	4873\$	4874	4876\$	4897	4909	4912	4919\$	4976
		4979\$	4984	5009\$	5032\$	5034	5039	5041\$	5045	5055	5057\$	5073	5103	5106
		5111	5174	5179\$	5182\$	5184\$	5186	5190\$	5195	5201	5209	5212	5219\$	5220
		5227\$	5236	5238\$	5257	5260	5264\$	5270	5274	5275\$	5278\$	5280\$	5288\$	5292\$
		5294	5296\$	5304\$	5306\$	5311	5345\$	5346	5362\$	5363	5368\$	5369	5384\$	5385
		5388\$	5389	5414	5419	5423\$	5428\$	5436\$	5438	5442\$	5452\$	5466	5470\$	5476\$
		5483\$	5534	5542	5544\$	5546	5558	5561\$	5563	5580\$	5600	5633\$	5649	5681
		5687\$	5689	5690\$	5693	5699\$	5704	5747	5751	5754	5758\$	5761	5765\$	5774\$
		5778\$	5783	5784	5786\$	5791\$	5802	5804	5807\$	5809\$	5812\$	5818\$	5821\$	5824\$
		5827\$	5831	5835	5842	5844	5846\$	5850	5853\$	5859\$	5865\$	5872\$	5876	5878
		5880\$	5883\$	5886	5897	5899\$	5902\$	5904\$	5907\$	5910\$	5911\$	5914\$	5917	5927\$
		5930\$	5932	5936\$	5937	5944\$	5950	5951\$	5953\$	5961	5965\$	5966	5976\$	5979\$
		5981\$	5983	6028	6030\$	6033\$	6035	6039\$	6041\$	6047\$	6048	6060\$	6063	6073
		6076	6079	6085\$	6088\$	6090\$	6093\$	6096\$	6099\$	6101	6109	6111\$	6116\$	6128\$
		6131\$	6133\$	6135	6151\$	6164\$	6168\$	6173\$	6177\$	6189\$	6196\$	6199	6207\$	6212
		6216	6220	6224	6276\$	6279\$	6281	6290\$	6293\$	6295\$	6297	6300\$	6302\$	6306\$
		6309\$	6311\$	6316\$	6323\$	6326\$	6328\$	6330\$	6332	6349\$	6352\$	6354	6358\$	6359
		6368\$	6374	6377\$	6379\$	6392\$	6394\$	6396\$	6404\$	6407\$	6409\$	6415\$	6419\$	6423\$
		6425	6434\$	6437\$	6440\$	6442\$	6445\$	6448\$	6467	6472	6478	6488	6490	
AACTIV	0040	177\$	2105	2124	2159	2272	2610	2701	3909	3958				
AAFLAG	0042	181\$	2271	2504	2507									
ABUFFR	71FE	2070	2103	2442\$	2458\$	2612	2703	3707	3719	3730	3981	4121	4212	6786\$

Sym'	Value	References (# = Definition, \$ = Write, <BLANK> = Read)													
ACFA10	5DFD	4545	4556#												
ACFA11	5E05	4560#	4562												
ACFA12	5E0C	3770	4563#	5234											
ACFA13	5E33	4582#	4585												
ACFA1L	5DES	1843	4543#												
ACHECK	4E2D	2245	2254#												
ACHK1	4E36	2263#	2274												
ACLO	4120	480#													
ACVECT	4C25	480	1843#												
ADDAD1	6251	1078	3536	4356	4924	4962	5016	5102#	6206						
ADDADR	624E	2997	3000	3003	3029	3032	3035	4998	5006	5020	5071	5101#			
ADRES1	7114	1946	6654#												
ADRES2	7145	2307	6677#												
ADRES3	716A	2077	6704#												
ADRSPC	71B5	2616	2707	2855	2879	2923	3070	6744#							
ADRTS1	4FBA	2478	2503#												
ADRTST	4FB2	2429	244#	2453	2499#										
AFLAG	0002	179#	2264	2967	3471										
AFTER	557C	3247	3267	3326#											
AFTER1	5597	3334	3336#												
ALARM	7090	6052\$	6054\$	6056\$	6058\$	6065	6163	6582#							
ALARM1	0007	233#	6088												
ALLOWP	0003	15#	5284												
ALRMM3	7094	5996	6070	6583#											
ALTCHK	4B2E	1663	1746#												
ALTFLG	4181	560#													
ALTMOD	001B	212#	1662												
ANDAD1	56AA	3489#	3496												
ANDADR	56A5	3479	3487#												
APTFLG	0004	19#													
APTLOD	40D6	419#	777\$	1834											
ARM12	0023	260#													
ARM45	0038	261#	6423												
ASLWR0	7492	3370	6993#												
ASMASK	000F	182#													
ASTRSK	4FA0	2475#	6712												
ATFLAG	0080	190#													
TFLOW	52DF	2605	2697	2959#											
TSIGN	4F9B	2472#	6711												
Reserved		503	508	694\$	1019\$	1025\$	1077\$	1122\$	1135\$	1168\$	1171	1185\$	1191\$	1209\$	
		1223\$	1224	1259\$	1410	1510\$	1523\$	1542\$	1557\$	1593	1654	1950\$	1960\$	1969	
		2006\$	2013\$	2081\$	2093\$	2094	2104	2106	2115	2117	2123	2158	2160	2173	
		2175	2217\$	2223\$	2242\$	2243	2247	2250	2263	2265	2269\$	2270	2287	2289	
		2302	2304	2466\$	2469\$	2472\$	2475\$	2483	2538	2540	2726\$	2752\$	2758\$	2781\$	
		2857\$	2858	2861\$	2869\$	2885\$	2889\$	2894\$	2899\$	2924\$	2925	2951	2996\$	2999\$	
		3002\$	3028\$	3031\$	3034\$	3041\$	3046\$	3050\$	3054\$	3057\$	3098\$	3099\$	3138\$	3234\$	
		3253\$	3388\$	3394\$	3397	3431\$	3469	3478\$	3489	3492\$	3514\$	3515	3518\$	3519	
		3522\$	3523	3525	3535\$	3612\$	3613\$	3614	3662\$	3779\$	3787\$	3878	3886	3887\$	
		3892	3924	3941\$	3971	4037\$	4045\$	4049\$	4057	4061	4094\$	4098\$	4122\$	4181\$	
		4228	4235	4288\$	4326\$	4327	4330\$	4332	4335\$	4337	4340\$	4342	4345\$	4351	

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)												
		4355\$	4366\$	4367	4370	4523\$	4610	4633\$	4686\$	4693\$	4699\$	4884\$	4923\$	4927\$
		4951\$	4961\$	4968\$	4974\$	4987\$	4997\$	5005\$	5015\$	5019\$	5028\$	5070\$	5091\$	5097\$
		5104	5107\$	5274\$	5280	5302	5366\$	5374	5414\$	5416	5434	5542\$	5549	5581
		5681\$	5690	5693\$	5697	5699	5940	5942\$	6066\$	6071\$	6082\$	6083	6109\$	6111
		6116	6205\$	6314\$	6321\$	6362	6366\$							
BACKSL	005C	171\$												
BASADR	74A7	4675\$	4703	4706\$	7008\$									
BASCON	74A9	4664	4742	7011\$										
BASE	007B	220\$												
BEFORE	555B	3226	3271	3286	3310\$									
BEGIN	0000	293\$	1801											
BLKFLG	7098	5180\$	5187\$	5881\$	5887\$	5900\$	5918\$	5928\$	5933\$	5977\$	5984\$	6031\$	6036\$	6086\$
		6102\$	6129\$	6136\$	6277\$	6282\$	6291\$	6298\$	6307\$	6333\$	6350\$	6355\$	6405\$	6416\$
		6426\$	6585\$											
BLKNUM	4174	542\$	1029	1068	1071\$	1153\$	1206	1208\$	1262\$					
BLOCK	4097	349\$	1030\$	1148\$	1296									
BMPPT	471D	1173\$	1179											
BOOT	576D	3627\$	6655											
BOOTEN	0007	27\$												
BOOTSW	0001	10\$	1646											
BPHALT	7222	3571\$	3746	4617\$	6822\$									
BPSUB	57D5	3640	3649	3664	3668\$									
BREAK	000D	223\$	5687											
BRKCNT	7221	6820\$												
BRKFLG	7220	5218	5683	6819\$										
BSPREP	57E4	1651	3684\$											
BUFADR	40CA	397\$	1637\$	1684	1694\$	1736	1738\$	1921\$	1932	1941	1944\$	1952\$	2038	2129
		2144\$	2379	2381\$	2440\$	2521	2524\$	2549	3627	3737	3817	4029		
BYTCNT	4095	348\$	1021\$	1283\$	1474	1478								
BYTE	4E92	2338\$	6677											
BYTE00	417A	551\$	1197	1204	1319	1327\$								
BYTE01	417B	552\$	1329\$											
BYTES	416E	539\$												
BYTSUM	4082	330\$	1041\$	1065	1639\$	3879	4116	4165	4173	4186	4198	4202\$	4203	4304
		4364\$												
C	Reserved	1515\$	1543\$	1550\$	1574\$	2218\$	2730\$	2738\$	2745\$	2749\$	2759\$	2763\$	2770\$	2774\$
		2856\$	2863	3242\$	3260\$	3376\$	3406\$	3412\$	3615	4028\$	4051\$	4059	4142\$	4146\$
		4152\$	4156\$	4709\$	4747\$	4765\$	5264	5270\$	5275	5306	5310	5436	5546\$	5550
		5561	5563\$	5571\$	5585	5588\$	5694\$	5696	5700	6315\$	6319\$			
CCFLAG	40CF	405\$												
CFAIL	6062	4875	4881	4887	4890	4892\$								
CFATAL	7565	4893\$	7119\$											
CFERR_*	0002	199\$	4412											
CFLOW_*	535B	2940	3046\$											
CFMSG	730A	4894	6869\$											
CF_ERR*	5D16	1976	4411\$											
CF_TL_*	738B	4413	6892\$											
CGIMOD	0001	240\$	6311											
CGFIVE	000D	280\$	6442											
CGFOUR	000C	278\$	6434											

Symbol	Value	References	(# = Definition, \$ = Write, <BLANK> = Read)
CHANGE	SDD7	4514	4519 4527#
CHECK *	652D	5535	5551 5557 5566#
CHKSUM *	00DE	310#	1401
CHRBUF	7001	1920	6520#
CHRCNT	7000	1918	4418\$ 6519#
CHRSIL	4B80	1812#	
CLDMSG	72EB	4865	6865#
CLDSTO	6027	3651	4868#
CLDSTI	602F	3666	4853 4871#
CLDSTR	6024	3641	4867# 5068
CLDSTX	6020	3689	4865# 5051 6469
CLKERR	5D70	2729	4457#
CLKFLG	4196	581#	992 2675 2727 2829 3146 3204 3217\$ 3332 3457 3595\$ 3710\$ 3743
		3904\$	4068\$ 4309 4452\$ 4615\$ 4735 4800\$
CLKMSG	73E2	4458	6904#
CLLTLK	4A46	1631#	1642
CLOCK *	0006	203#	4457
CLRACK	00A9	147#	1560\$ 2778\$ 4449\$ 5622\$ 5635\$ 5647\$ 5659\$ 5665\$ 5671\$ 5678\$ 5707\$ 5714\$
		5722\$	5736\$ 5743\$ 5782\$ 5815\$ 5841\$ 6153\$ 6226\$ 6231\$ 6458\$ 6461\$ 6464\$ 6471\$
		6477\$	
CLRACL	0033	73#	
CLRATN	00AB	157#	2744\$ 4151\$ 4448\$ 4763\$ 5619\$ 5713\$ 5721\$ 5729\$ 5735\$ 5744\$
CLRBLK	003A	84#	4552\$ 4559\$ 4572\$ 5188\$ 5888\$ 5919\$ 5934\$ 5985\$ 6037\$ 6103\$ 6137\$ 6283\$
		6299\$	6334\$ 6356\$ 6427\$
CLRBSY	002E	66#	
CLRCLD	6AE7	6471#	6635
CLRCLK	0020	45#	975\$ 2662\$ 2837\$ 3132\$ 3215\$ 3311\$ 3450\$ 3586\$ 3703\$ 3902\$ 4261\$ 4450\$
		4578\$	4613\$ 4656\$ 6465\$
CLRCCK	0024	51#	1522\$
CLRCGR	0038	81#	1513\$
CLRDCL	0031	70#	
CLRHLT	00AF	163#	1996\$ 3559\$ 5745\$
CLRLIT	003C	87#	2628\$ 2719\$ 4577\$ 4642\$ 5746\$ 6466\$
CLRMCI	0029	58#	
CLRMCK	002B	61#	
CLRPAG	003F	91#	
CLRPCR	6994	6268#	
CLRPFI	00AD	160#	3560\$ 4576\$
CLRPME	6AF8	6482#	6637
CLRSS	0022	48#	2843\$ 3445\$ 3913\$ 3920\$ 4291\$ 4760\$ 4762\$ 4770\$ 4772\$
CLRTIM	00A5	141#	3565\$ 5858\$ 6273\$
CLRTMR	002C	63#	3561\$ 3760\$ 4563\$ 5868\$ 6026\$ 6272\$ 6391\$
CLRUPO	00AB	153#	1547\$
CLRUPK	0026	54#	1556\$ 1600\$ 4279\$
CLRWCK	0036	78#	1598\$ 4277\$ 4403\$
CLRWRO	748C	3366	6985#
CLRWWM	6AF1	6477#	6634
CMDVAL	4182	561#	1671 1714\$ 1746 4417\$
CMPADR	4162	528#	1046\$ 1281\$ 1389 1394 1442\$
CMPARI	7102	6652#	

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)															
CMPAR2	7139	6676#															
CMPAR3	715E	6701#															
CMPCH1	6244	5092#	5098														
CMPCHK	6242	4214	4218	5065	5088	5091#											
CMPTB1	4183	563#	1045	1280													
CMPTB2	4185	566#	1441														
CNSOLA	4C42	1909	1913#														
CNSOLB	4C5E	1910	1926#	1937	4818	4828	4888										
CNSOLX	4C7B	1922	1934	1940#	2041												
CNTADR	40C8	396#	829	860	1635\$	1670	1687	1725	1750								
CNTINO	58E2	3750	3808#														
CNTIN1	586B	3745	3755#														
CNTIN2	5877	3759#	3809	5043													
CNTRLR	42AC	821#	1661														
CNTRLR1	42BD	826	828#														
CNTRLU	42CE	857#	1659														
CNTRLC	0003	215#															
CNTRLO	000F	214#															
CNTRLP	0010	219#	5282														
CNTRLQ	0011	218#	5588														
CNTRLR	0012	213#	1660														
CNTRLS	0013	217#	5571														
CNTRLU	0015	216#	1658														
CNTRLX	0018	221#															
CNVER1	4E0A	2196	2202#														
CNVERT	4DF4	2119	2139	2192#	2369	2403	2447	2571									
CODFIL	7247	4097	4826	6838#													
CODFL1	4C2E	778	1846#														
CODFLG	408B	337#	695	716	779\$	824	1649	1784	4024								
CODTAB	7450	2854	6936#														
COLD	408A	336#	3670\$	3685\$	4073\$	4801\$	4872	4877\$	6473\$								
COLD_B*	5F99	1836	4076	4797#													
COLD_B*	5FD2	4078	4823#														
COLD_B*	5FE8	4079	4833#														
COLD_H*	601A	4838	4860#														
COLON	003A	175#															
COMERR	5D5E	2740	2751	2765	2776	3100	4160	4169	4189	4448#	5608	6238	6250				
COMFIN	568A	3043	3047	3051	3055	3058	3469#										
COMMN1	4E9E	2339	2341	2343#													
COMMON	4E8C	2319	2321	2323	2325	2327	2329	2331#									
COMMSG	7378	4454	6888#														
COMM_C*	0005	202#	4453	4479													
COMWR0	748F	3368	6989#														
CONSMS	1000	192#															
CONBUF	74AD	4667	7014#														
CONHL1	58CA	3796#	4601														
CONHL2	58A3	3774	3778#														
CONHL3	58C1	3777	3791#														
CONHLT	587A	3760#															
CONMOD	71FD	3756\$	5035\$	6784#													

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)													
CONMSG	41DF	624#	723												
CONPAK	71FC	3757	5036	6783#											
CONSI2	74A8	4679\$	4685	7009#											
CONSOL	41CE	620#	693												
CONT	5840	3737#	6656												
CONT0	5848	3728	3741#	3962	3968										
CONTLR	41BE	612#	821												
CONTLU	41BA	610#	857												
COPY	00F9	311#	967	1035	1097	2415	2608	2614	2667	2705	2799	2961	3302	3363	
		3455	3724	3983	4103	4666	4745	4918	4939	4944	4957	4991	4996	5013	
		5023	6187												
COPY1	00FB	312#	699	1260	3143	3235	3254	3389	3432	3663	4095	4123	4289	4700	
		4885	4928	4952	4988	5029									
COPYB0	419C	590#	1083\$	1102											
COPYB1	419D	591#													
COUNT	4187	569#	986\$	1016	1101\$	1588									
COUNT1	7101	1945	6651#												
COUNT2	7138	2306	6675#												
COUNT3	715D	2076	6700#												
CPATTN	0083	124#	2760	2771	3101	4088	5229	5313	5335	5357	5488	5509	5604	6233	
		6245													
CPCH0	62BD	5207	5210	5213	5218#										
CPCH1	62D0	5221	5224	5229#											
CPCHK1	629D	5205#	5515	5519	5521	5530									
CPCHK2	6375	5263	5266	5272	5290	5299	5313#	5560	5568						
CPCHK3	639A	5319	5323	5325	5335#										
CPCHK4	63BE	5343	5347	5357#											
CPCHK5	64A9	5390	5418	5432	5450	5465	5478	5481	5488#						
CPCHK6	64CB	5496	5499	5509#											
CPFLAG	40F3	435#													
CPSER1	6560	5604#	5609												
CPSER2	656D	5606	5610#												
CPSERV	6558	3104	5231	5315	5337	5359	5490	5511	5599#						
CPUACK	0082	122#	2735	2746	4143	4153									
CR	000D	208#	610	612	618	624	626	704	1701	1711	1748	1958	2363	2399	
		2535	2561	2581	3629	3638	3739	3781	3819	4036	4053	4430	4599	4628	
		4856	6653	6701	6832	6836	6839	6842	6845	6848	6851	6853	6855	6857	
		6859	6863	6865	6867	6871	6873	6875	6877	6882	6886	6890	6894	6898	
		6902	6906	6911	6918	6932									
		4844	6847#												
CRAFIL	7279	4384#	6669	6670	6671										
CRCMD	5D03	198#	4462	4482											
CRD_CO*	0001	4461#	4840												
CRD_ER*	5D78	4463	6908#												
CRD_ER*	73FA	441#	1793	4058\$	4834										
CRD_FL*	40FD	513	1779#												
CRD_RD*	4B3C	316#													
CRD_VE*	0398	2537#	6704												
CRFLOW	4FD6	4846	6850#												
CRMFIL	728B	128#													
CSR07	0085														

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)												
CSR15	0086	130#												
CSK23	0087	132#	1516											
CSRBUF	41A4	596#	3320	3328										
CTLDIS	40FE	443#	775\$	4113\$	4177\$	4374\$	4428\$	4474\$						
CVECTR	4C28	1844#												
D	Reserved	693\$	698\$	966\$	970\$	1034\$	1069\$	1070	1096\$	1184\$	1186	1190\$	1207	1333\$
		1337\$	1351	1671\$	1672	1675	1701\$	1946\$	2011\$	2012\$	2019	2021\$	2022	2071\$
		2077\$	2133\$	2138	2307\$	2414\$	2606\$	2612\$	2664\$	2703\$	2798\$	2867\$	2870\$	2960\$
		3137\$	3230\$	3233\$	3245\$	3252\$	3263\$	3275\$	3277\$	3280\$	3293\$	3301\$	3326\$	3349\$
		3366\$	3368\$	3370\$	3374\$	3387\$	3403\$	3420\$	3453\$	3661\$	3719\$	3723\$	3778\$	3786\$
		3856\$	3940\$	3981\$	4093\$	4097\$	4102\$	4121\$	4212\$	4217\$	4287\$	4413\$	4420\$	4435\$
		4440\$	4445\$	4454\$	4458\$	4463\$	4515	4516\$	4520	4521\$	4611	4632\$	4665\$	4684\$
		4689	4692\$	4695	4698\$	4711\$	4741	4742\$	4743	4883\$	4917\$	4926\$	4938\$	4943\$
		4950\$	4956\$	4967\$	4969	4973\$	4986\$	4990\$	4995\$	5012\$	5022\$	5027\$	5064\$	5087\$
		5090\$	5092	5096\$	5294\$	5298	5320\$	5321	5327	5329	5330\$	5331	5333	5339\$
		5340	5349	5351	5352\$	5353	5355	5412\$	5415	5420	5421\$	5424	5462\$	5463
		5467	5468\$	5471	5492\$	5493	5501	5503	5504\$	5505	5507	5516\$	5517	5523
		5525	5526\$	5527	5529	5610\$	5611	5612\$	5616\$	5617	5652	5654	5660	5662
		5666	5668	5672	5674	5695	5708	5710	5715	5717	5723	5725	5730	5732
		5737	5739	5770	5772	5798	5800	5844\$	5862	5865	5872	5986\$	5988	5991
		5994	5999	6003\$	6005	6008\$	6010	6065\$	6070\$	6081\$	6186\$	6363	6365\$	6487\$
		6492\$												
DACTIV	0080	176#	2116	2174										
DARM12	0083	272#												
DATCHK	6233	4931	4935	5086#										
DATLEN	71B4	2882	2977\$	2989	3021	3061	6743#							
DBUFFR	7202	2114	2606	4217	6787#									
DCHK1	4E28	2247#	2258											
DCLO	0002	12#												
DDECOD	4FDE	2549#	6657											
DECOD1	4CFB	2007#	2015	2082										
DECODE	4CFA	1947	2006#	2084	2308									
DECODX	52AE	2615	2706	2923#										
DEFDRV	4150	515#	781\$	2558	3868	4812	4847	4867						
DEFFIL	7255	4869\$	4870	6841#										
DELAY1	0457	193#												
DELETE	007F	211#	1664											
DELFLG	40CE	403#	835\$	1667	1719\$	1730								
DELSK	00FE	189#												
DEPADR	71D2	6761#												
DEPCOD	0001	185#	2602	3183	6923	7001								
DEPDAT	71D6	6762#												
DEPIO	50B6	2647	2654#											
DEPLS1	54C9	3246#	3278	3298										
DEPLS2	54CC	3247#	3303											
DEPPAK	71D0	4986	6760#											
DEPWR0	5543	3292	3300#											
DEST	7479	3285\$	3296\$	3395	6958#									
DEVFLG	7212	2145	3646	3862\$	6799#									
DFAULT	52D5	2927	2948#											

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)											
ENDSE1	4787	1216	1220	1226#									
ENDSEG	476D	1157	1214#										
ENTER	625F	3759	5173#										
ENTYPE	4178	546#	1158\$	1163	1193								
ERROR	5D80	4414	4421	4436	4441	4446	4455	4459	4464	4466#			
ERROR1	5DA2	4432	4471	4481#									
EXAM11	511F	2702	2706#	2723									
EXAM12	5122	2707#	2720										
EXAMIN	50F2	2689#	6658										
EXAMQ	5511	3274	3280#										
EXE1	5663	3439	3443#										
EXECUT	5649	3246	3264	3350	3367	3369	3375	3421	3429#	4712			
EXMCOD	0000	184#	3140	3154	3213	3228	3273	3288					
EXMIO	51F1	2806	2813#										
EXMLS	54D1	3229	3251#										
EXMLS1	54EC	3264#	3281	3308									
EXMPAR	5435	3155	3175#										
EXMWR	554F	3289	3305#										
EXTBUF	410E	471#	3852	3858									
EXTCNT	7211	2163	3654	6798#									
EXTFLW	4DDA	2113	2163#										
EXTPTR	720E	2169	2172\$	3853\$	6796#								
EXTRAS	42FC	875#	1169										
EXTYPE	523A	2714	2854#										
FASTWR	5F4D	4696	4752#										
FASUPC	0016	230#	4753										
FASWR1	5F65	4761#	4776										
FASWR2	5F78	4767	4769#										
FIXCP1	46B3	1118	1121#										
FIXCPU	46A6	1089	1114#	1124									
FIXER1	466B	1067	1090#										
FIXER2	4685	1094	1099#										
FIXERR	4663	1053	1059	1087#									
FIXUP	5A0D	3911	3960	3979#									
FLAGS	7213	1983\$	2093	2242	2269	2477	2499	2598	2609	2690	2700	2966	3470 3841
		3907	3956	3992\$	6802#								
FLAGS1	7214	1984\$	2603	2695	2969	3473	3720	6803#					
FLAGS2	7215	1985\$	2352	2367	2374	3514	3979	6804#					
FLAGS3	7216	1986\$	2384	2401	2408	2418	6805#						
FLIP	543D	3125	3182#										
FLIP1	547F	3206	3208#										
FNBUF	4107	469#	692	740	1184	3661	3854						
FNCNT	7210	2148	3643	3845	3865\$	6797#							
FNDFLG	4164	531#	937\$	958	962	1199							
FNFLOW	4D8D	2102	2123#										
FNFLW1	4DC0	2125	2128	2131	2137	2148#							
FNFLW2	4DD3	2147	2158#										
FNFMSC	41EF	628#	800										
FNPTR	720C	2154	2157\$	3855\$	6795#								
FOOFIL	7267	3659\$	3660	3665	6844#								

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)												
FOUR	SCCA	4342#												
FOURDN	5343	3023	3028#											
FOURUP	5317	2991	2996#											
FU_VEC*	004F	302#												
GCVECT	4129	485#												
GET512	47E0	1250	1274#											
GETCHK	4CC6	1953	1962	1974#										
GETCS	5B67	4167	4168	4171#	4172	4223	4282							
GETDA1	5CCF	4347#	4348											
GETDAT	5CAC	4138	4149	4158	4222	4267	4270	4273	4316#					
GETEND	492F	1054	1290	1439#										
GETL10	4A58	832	836	1633	1640#	1653	1712	1728	1741					
GETL11	4A7A	1654#												
GETL12	4AA0	1670#	1722											
GETL13	4AAD	1674	1677#											
GETL14	4A91	1657	1664#											
GETL15	4ABC	1679	1681	1683#	1749									
GETL16	4AD3	1690	1695#											
GETL17	4AF5	1700	1702	1704	1709#									
GETL18	4AF6	1697	1710#											
GETLIA	4A74	1645	1648	1652#										
GETLIN	4A4D	483	1635#	1919										
GETNXT	4FC5	2075	2305	2355	2360	2387	2392	2430	2446	2454	2521#	2534		
GETON1	47A8	1241	1245#											
GETON2	47AE	1235	1248#											
GETON3	47D4	1256	1267#											
GETON4	47D7	1238	1268#											
GETONE	478C	1154	1155	1173	1232#	1326	1328							
GETTWO	484A	1196	1203	1318	1326#									
GETUPC	59B6	3914	3921	3933#	4480	4630								
GFLOW	5355	2933	3041#											
GLVECT	4123	482#												
GOSV	4103	460#												
GO_STA*	4C00	784	1829#											
GPR	4E76	2322#	2459	6684										
GPRCON	7458	3041	6938#											
GRPCNT	74A6	4672#	4713	4715#	7007#									
GSVECT	4145	506#												
GS_VEC*	0046	299#	507	1652	3918	3966	4114	4171	4347	5262	5431	5559		
GT512A	47E5	1276#	1289	1291										
H	Reserved	692#	707#	710#	716#	723#	735#	740#	782#	800#	821#	823#	827#	857#
		944#	951#	953#	965#	981#	988#	999	1002#	1004#	1009	1033#	1036#	1045#
		1065#	1075#	1095#	1105	1110#	1115#	1116	1132	1134#	1147#	1166#	1175#	1177
		1183#	1189#	1199#	1219	1221#	1232#	1248#	1251#	1255#	1258#	1270#	1274#	1278#
		1280#	1282#	1285#	1296#	1305	1307#	1312#	1314#	1316#	1346#	1349	1353	1358#
		1379#	1395#	1400#	1407	1411#	1413#	1417#	1421#	1425#	1427	1433#	1439#	1441#
		1444	1445#	1447#	1452#	1456#	1460#	1464#	1468#	1470#	1473#	1477#	1481#	1483#
		1486#	1497	1498#	1500#	1502#	1505#	1573#	1588#	1604#	1636#	1693#	1730#	1737#
		1782#	1784#	1795#	1832#	1916#	1918#	1920#	1929#	1943#	1945#	1967#	1974	1978#
		2007#	2017#	2023#	2034#	2056#	2070#	2076#	2103#	2114#	2132#	2134#	2143#	2145#

Symbol Value References (# = Definition, \$ = Write, <BLANK> = Read)

		2148\$	2156\$	2163\$	2171\$	2219	2222\$	2306\$	2358\$	2367\$	2370\$	2374\$	2380\$	2390\$
		2401\$	2404\$	2408\$	2413\$	2418\$	2439\$	2477\$	2482\$	2499\$	2523\$	2553\$	2557\$	2565\$
		2569\$	2575\$	2579\$	2601\$	2607\$	2613\$	2665\$	2670\$	2698\$	2704\$	2725\$	2732\$	2742\$
		2756\$	2757	2769\$	2780\$	2783\$	2797\$	2838\$	2854\$	2862\$	2866\$	2873\$	2896\$	2898\$
		2905\$	2909\$	2923\$	2959\$	3061\$	3063\$	3073\$	3075\$	3079\$	3087\$	3113\$	3136\$	3199\$
		3220\$	3232\$	3236\$	3237\$	3240\$	3241	3251\$	3256\$	3258\$	3259	3265\$	3300\$	3313\$
		3316\$	3318\$	3320\$	3322\$	3328\$	3330\$	3354\$	3362\$	3377\$	3385\$	3386	3398\$	3399\$
		3401\$	3411\$	3429\$	3433\$	3435\$	3437\$	3451\$	3454\$	3487\$	3493\$	3506\$	3526\$	3528\$
		3530\$	3532\$	3537\$	3553\$	3567\$	3579\$	3581\$	3590\$	3598\$	3660\$	3665\$	3707\$	3718\$
		3725\$	3746\$	3757\$	3766\$	3771\$	3783\$	3785\$	3789\$	3821\$	3823\$	3852\$	3854\$	3858\$
		3860\$	3863\$	3879\$	3882\$	3888\$	3891\$	3907\$	3933\$	3936\$	3939\$	3943\$	3954\$	3956\$
		3979\$	3982\$	3994\$	3999\$	4002\$	4024\$	4035\$	4043\$	4092\$	4096\$	4101\$	4120\$	4133\$
		4165\$	4180\$	4197\$	4213\$	4216\$	4225	4229\$	4245	4250\$	4263\$	4286\$	4306\$	4316\$
		4318\$	4320\$	4322\$	4331\$	4336\$	4341\$	4346\$	4354\$	4361\$	4365\$	4395\$	4398\$	4401\$
		4467\$	4498\$	4522\$	4581\$	4582\$	4583	4592\$	4612	4631\$	4645\$	4658\$	4661\$	4664\$
		4667\$	4673\$	4676\$	4680\$	4681	4682\$	4691\$	4697\$	4701\$	4702\$	4707\$	4708	4717\$
		4721\$	4724\$	4728\$	4730\$	4744\$	4746\$	4753\$	4755\$	4764\$	4788\$	4816\$	4826\$	4834\$
		4844\$	4846\$	4850\$	4851\$	4852\$	4865\$	4870\$	4872\$	4882\$	4892\$	4894\$	4907\$	4916\$
		4922\$	4925\$	4937\$	4942\$	4945\$	4949\$	4953\$	4955\$	4960\$	4963\$	4966\$	4972\$	4978\$
		4985\$	4989\$	4994\$	5001\$	5011\$	5014\$	5021\$	5026\$	5036\$	5047\$	5049\$	5053\$	5063\$
		5067\$	5086\$	5089\$	5095\$	5101\$	5108\$	5189\$	5214\$	5216\$	5218\$	5222\$	5225\$	5291\$
		5295\$	5301\$	5303\$	5361\$	5365\$	5367\$	5373\$	5383\$	5387\$	5422\$	5427\$	5433\$	5435\$
		5451\$	5469\$	5475\$	5482\$	5485\$	5537\$	5540\$	5543\$	5547\$	5553\$	5574\$	5579\$	5584\$
		5591\$	5593\$	5603\$	5650\$	5683\$	5685\$	5749\$	5773\$	5777\$	5785\$	5790\$	5804\$	5812
		5829\$	5845\$	5935\$	5943\$	5964\$	5989\$	5992\$	5996\$	6000\$	6004\$	6009\$	6046\$	6059\$
		6062\$	6064\$	6091\$	6094\$	6097\$	6150\$	6163\$	6167\$	6172\$	6176\$	6182\$	6185\$	6188\$
		6190\$	6192\$	6194\$	6213\$	6217\$	6221\$	6274\$	6313\$	6318\$	6325\$	6328	6340	6357\$
		6367\$	6386\$	6412\$	6414\$	6421\$	6439\$	6444\$	6447\$	6489	6491\$	6494\$		
HALT	58E8	3817\$	6659											
HALTPC	71DC	3785	4990	6766\$										
HEXASC	5DBD	2868	2871	3780	3788	3942	4099	4508\$	4524					
HIGHST	42FA	874\$												
HLBUFF	4088	334\$												
HLTBUF	71DA	3766	6764\$											
HLTCOD	71DB	3771	3799	4598\$	5003	6765\$								
HLTFLG	0002	13\$	3796	4861										
HLTMSG	7414	3778	3783	3821	6913\$									
HLTPSL	71E0	4995	6767\$											
HMPEND	707F	3765\$	3732\$	4594	6558\$									
HOLD1	0019	246\$	5904	5981	6133									
HOLD4	001C	251\$	6409											
ICCS	707C	3572\$	5189	5845	5890	5923	5943	6067	6108	6125	6345	6367	6553\$	
ICHECK1	53AC	3099\$	3103											
ICHECK	53A2	2639	2795	3094\$										
ICON	745C	3054	6940\$											
IDLE	4C32	1837	1844	1845	1906\$	1911	3798	3801	4483	4648	4787	4819	4829	4958
		4863	4899	6266	6269									
IFLAG	7223	1995\$	3073	5834	6824\$									
IFLOW	5367	2937	3054\$											
IMPXFR	7099	3576\$	5879\$	5939\$	6029\$	6150	6361\$	6587\$						

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)													
INCCNT	417F	550#	1172#	1174	1176#										
INDADR	721C	1951	1960#	4001#	4017#	4027#	4071#	6813#							
INDBUF	7600	3999	6013	7133#											
INDCNT	721B	1931	1971#	6012#											
INDFLG	409D	355#	1907	1956#	4002	4427#	4476#	4015#	4025#	4079#	5177#				
INDRC1	5A23	3994#													
INDRCT	5A1B	3991#	6654												
INDTS1	4CA7	1955	1950#	1963											
INDTS2	4CB6	1959	1964#												
INDTST	4C94	1926	1950#												
INISUB	56F0	3552	3559#	3717	4880	4914									
INIT	56E6	3552#	3557	6660											
INITH	0035	76#	3583#	6455#											
INITL	0034	75#	3585#	6457#											
INITPK	71FB	3590	6701#												
INIVEC	0040	297#	4790												
INTACK	6575	5616#	6601												
INTCSR	7074	3593#	5214	5361	5376#	5383	5760	5766#	6545#						
INTDB	7075	5763	6546#												
IOARG1	50BF	2655#	2657#												
IOARG2	51F7	2014#	2015#												
IPR	7071	5829	6540#												
IPRCHK	66B6	5203	5819	5822	5825	5829#									
IPEG	4E80	2326#	6685												
ISR	7070	3568#	5320	5339	5492	5516	5612	6539#							
ITCSR	001D	34#	4551#	4558#	4569#	4571#	5183#	5185#	5884#	5903#	5905#	5912#	5915#	5931#	
		5980#	5982#	6034#	6089#	6132#	6134#	6280#	6294#	6296#	6310#	6312#	6324#	6331#	
		6353#	6393#	6408#	6420#	6424#	6435#	6443#							
ITDB	001C	33#	5906	5909	5987	5990	5993	5995	6092#	6095#	6098#	6100#	6138	6141	
		6144	6147	6169	6284	6287	6317#	6327#	6329#	6438#	6441#	6446#	6449#		
ITEST	537D	2637	2793	3070#											
ITEST1	538B	3075#													
ITEST2	539C	3084	3087#												
ITICR	68B2	6122#	6624												
ITICR0	68C1	6124	6128#												
ITICR1	68CE	6134#	6410												
ITICR2	68FD	6127	6163#												
ITICR3	68ED	6148#	6178												
ITNICO	6947	6195	6198	6200	6205#										
ITNIC1	694D	6201	6207#												
ITNICR	691B	6182#	6625												
ITRCSR	689B	6108#	6622												
ITRUN	679E	5867	5958#												
ITRUN0	67F4	5968	5998	6002	6007	6012	6016#								
ITRUN1	67FF	5959	6026#												
ITRUN2	67F6	6017#	6043												
ITSGI	707E	3575#	5893#	5964	6555#										
ITTEMP	709E	5986	6591#												
ITWCS0	66E6	5852	5855	5858#											
ITWCS1	66E8	5857	5859#												

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)												
ITWCS2	671F	5874	5890#											
ITWCS3	6747	5908	5911#											
ITWCS4	6752	5913	5916#											
ITWCS5	6768	5925	5927#											
ITWCS6	6789	5938	5943#											
ITWCS7	6793	5946	5948#											
ITWCSR	66C6	5840#	6623											
L	Reserved	1000	1010	1103	1117	1129	1131\$	1178	1218	1302	1304\$	1428	2020\$	3864\$
		4233	4255	4584	5600\$	5802\$	5809	6326						
LA1245	007B	263#	6330											
LARM12	0063	265#	5883	5930	6033	6392								
LARM2	0062	270#	6352											
LARM3	0064	268#	4550	4557										
LARM45	0078	266#												
LAST	6590	5634#	5641	5644	6112	6117	6119							
LCMASK	00DF	188#	1682											
LDCSR	49A4	1504	1509#	3321	3329	4307								
LDCSR1	49A7	990	1510#	2672	2844	3144	3202	3443	4626					
LDUPC	49D1	982	989	1111	1540#	2839	3221	3319	3331	3452	3591	3708	3934	4264
		4726	4756											
LDUPC1	49D4	1541#	2668	2673	2846	3198	3201	3456	3938	4308				
LDXPRP	4500	477	700	932#	3834									
LENGTH	4166	535#	968	1198\$										
LF	000A	209#	624	1703	1965	6836	6839	6842	6845	6848	6851	6859	6867	6873
		6875	6877											
LIMIT	41B4	603#	1151\$	1214	1239									
LIMSP	41B5	604#	955\$	1242										
LIST	46D9	961	1147#											
LIST1	46DC	1148#	1225											
LIST2	4710	1165	1168#											
LIST3	4713	1169#	1210											
LIST4	472F	1162	1183#											
LIST5	4737	1186#	1192											
LIST6	4759	1188	1195	1203#										
LIST7	46EC	1154#	1180											
LIST8	46E3	956	1151#											
LNGSAV	71E8	4942	4967	6771#										
LNGSUM	71EC	4945	4960	4966	6773#									
LOAD	58FC	3833#	6661											
LOAD1	45C3	1005	1011	1016#										
LOAD12	0043	257#												
LOAD1X	45F9	1036#	1098	1112										
LOAD2X	4625	1060#												
LOAD3X	463A	1062	1068#											
LOAD45	0058	258#	6419											
LOAD4X	4656	1076	1080#											
LOADA	45A1	974	987	998#	1006									
LOADB	45B6	1001	1008#											
LOADED	7082	5895	6054	6113	6122	6571#								
LOADX3	454E	960	965#											

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)													
LOADZ	58FF	3834#	4004												
LODADR	40A3	362#	965	978	1034	1075	1081\$	1095	1107	1277\$	1406	1412\$	1443	1446\$	
		1487\$													
LODCNT	40A7	364#	1387\$	1400	1413	3883									
LODDST	4165	533#	1044\$	1278	1402	1439									
LODFLG	409B	353#	939\$	1013\$	1142\$	1160	1990\$	2073	2100	2111	2256	3867\$	4426\$		
LODSU1	5925	3642	3840	3852#											
LODSUB	590A	3833	3840#	3993											
LODVEC	4117	477#													
LOGCNT	7E0F	7158#													
LOGCON	7E0E	7157#													
LONG	4E9C	2342#	6679												
LPCNT1	7476	3365\$	3379	3382\$	3419\$	3422	3425\$	6954#							
LPCNT2	7477	3347\$	3356	3359\$	6955#										
LRMASK	40BE	385#	1643	1695	3775	5391									
LSHIFT	4A00	1518	1545	1552	1570#	1575	2221	3244	3262	3378	4710	4748	4766		
LSNWRN	747A	3252	6961#												
LSTBUF	41C2	614#	1166	1183	1312	1314	1316								
LSTPRP	46CB	478	1141#	2583											
M	Reserved	708\$	709	717	719\$	783\$	830	861\$	945	1066	1187	1200\$	1233	1245	
		1252\$	1253	1269	1275\$	1279\$	1297\$	1378\$	1390	1396	1410\$	1414\$	1418	1422	
		1434\$	1440\$	1453	1457	1462	1466	1469	1471	1475	1479	1482	1484	1499\$	
		1501\$	1503\$	1570	1572\$	1589	1591\$	1602\$	1676\$	1685\$	1688	1691\$	1727	1729\$	
		1731	1733\$	1739	1785	1788\$	1796	1964	1977	2006	2008	2035\$	2036\$	2039	
		2053	2130	2133	2136	2146\$	2149	2153\$	2155\$	2164	2168\$	2170\$	2226	2227\$	
		2368\$	2375	2378\$	2402\$	2409	2412\$	2436	2481\$	2483\$	2500	2503	2506	2508\$	
		2522	2550	2554	2566	2570	2576	2602\$	2650\$	2731	2741	2768\$	2779\$	2784	
		2808	2859	2897\$	2907	2910	2928\$	2948	3062	3064	3074\$	3076	3078\$	3080	
		3088	3090\$	3239\$	3257\$	3314	3315\$	3317\$	3353\$	3400\$	3408	3410\$	3434	3436	
		3438	3440	3442\$	3490	3491\$	3527	3529	3531	3533	3628	3747	3749\$	3772	
		3880	3881\$	3886\$	3892\$	3908	3955\$	3957	3980\$	3995	3998\$	4003\$	4025	4027\$	
		4042	4166	4199	4228\$	4248	4317	4319	4321	4323	4328	4329\$	4333	4334\$	
		4338	4339\$	4343	4344\$	4366	4369\$	4393	4396	4399	4508	4517	4659	4660\$	
		4662\$	4668	4674	4677	4690\$	4704\$	4723\$	4725\$	4729	4731\$	4835	4836\$	4849\$	
		4873	4954\$	4964\$	4970	4979	4984\$	5093	5105	5106\$	5190	5195\$	5219	5223\$	
		5226\$	5292	5296	5302\$	5304	5309	5311\$	5345	5362	5366	5368	5371\$	5374\$	
		5384	5388	5423	5428	5434\$	5438\$	442	5452	5470	5476	5483	5486\$	5538	
		5541	5544	5556	5558\$	5575\$	5580	5583\$	5585\$	5592\$	5594\$	5633	5649\$	5651\$	
		5684\$	5686\$	5703	5704\$	5774	5778	5784\$	5786	5789\$	5791	5794\$	5830	5831\$	
		5846	5849	5850\$	5853	5936	5944	5949	5950\$	5965	5967\$	5997	6001	6006	
		6011	6047	6050\$	6060	6061\$	6063\$	6083\$	6090	6093	6096	6099	6151	6154\$	
		6164	6168	6173	6177	6189	6191	6193	6196	6212\$	6216\$	6220\$	6224\$	6316	
		6358	6368	6373	6374\$	6437	6440	6445	6448	6490\$					
MAKCN1	SEB8	4668#	4718												
MAKCN2	SECC	4682#	4716												
MAKCN3	SEDC	4689#	4694												
MAKCN4	SF16	4671	4720#												
MAKCN5	SF31	4733	4735#												
MAKCN6	SF39	4688	4741#												
MAKCN8	SEES	4696#	4749												

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)			
MAKCON	5E9E	3597	4589	4656#	4805
MAKEWR	5604	3297	3307	3385#	
MAKMOD	5373	2636	2792	3061#	
MAKSPC	5293	2886	2890	2896#	
MARKSP	5DB1	1940	4496#	5077	5082
MASTER	0017	236#	6279	6323	
MATCH	4D07	2010	2017#		
MAXCNT	0050	194#	1689		
MCHECK	53BA	2635	2791	3113#	
MCON	7460	3057	6942#		
EMADR	749E	4092	7003#		
MEMCOD	74B4	4101	7022#		
MEMDEP	508F	2618	2634#		
MEMERR	5D46	2787	4434#		
MEMEXM	51C1	2709	2790#		
MEMLD1	5971	3885	3891#		
MEMLOD	5957	1404	3878#		
MEMMSG	7365	4435	6884#		
MEMSIZ	74A2	5063	7004#		
MEM_CO*	0004	201#	4434		
MERGE	52FB	2974#	3480		
MFLOW	536D	2935	3057#		
MFLOW1	542A	3158	3167#		
MICCON	580C	3709#	3754	3806	3808
MICDEP	50A5	2620	2645#		
MICEXM	51E0	2711	2804#		
MICFLG	721F	1991\$	3906\$	4034	6817#
MICMON	4154	519#	698	4060\$	
MICPRO	4207	633#	827		
MICRO	4E85	2328#	6686		
MICST1	598F	3912#	3916		
MICST2	599C	3918#	3927		
MICST3	599F	3910	3919#		
MICST4	59A6	3922#	3923		
MICSTP	5976	3901#	6662		
MINCNT	708C	6081	6274	6579#	
MINUS	4F96	2469#	6710		
MIPFLG	40D9	424#	1795	5256	
MISC2	0001	9#			
MKSPC1	5296	2897#	2900		
MMSTRT	4C00	284#			
MODADR	4099	351#			
MODIFR	4090	344#			
MODVEC	0013	295#			
MREG	4E7B	232#	2443	6683	
MRKBUF	7219	4486	4500\$	6810#	
MSGADR	40B7	376#	727		
MSGBUF	40F0	433#			
MSGCNT	408E	342#			
MSGTBS	7E07	7150#			

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)											
MSKCPY	40BF	386#											
MSTRCD	0A0C	238#	6325										
MUL2	4823	1170	1222	1301#									
MULSH1	46C0	1129#	1136										
MULSHF	46BE	1020	1026	1127#									
MYCNT	4179	548#	689\$	707	720								
MYLDER	4266	706	727#	4429									
MYLOAD	420C	459	688#	1797									
MYPC	40B9	378#	4468\$										
NACKCM*	0007	204#	4439										
NACKDA*	0008	205#	4444										
NACK_X*	5D4E	4118	4438#										
NACK_X*	5D56	4175	4443#										
NAME\$	4302	885#											
NBUFFER	7208	2358	2370	3526	3982	6793#							
NEG1	7196	3034	3535	4326	4355	6727#							
NEG2	719A	3031	6729#										
NEG200	71A6	4917	6735#										
NEG3	719E	6205	6731#										
NEG4	71A2	3028	6733#										
NEWAS	71B3	1993\$	2331\$	2924	3700	3848	4126	6741#					
NEWCN0	6874	6077	6081#										
NEWCN1	6878	6069	6074	6080	6083#								
NEWCN2	687A	6085#	6275										
NEWCNT	684C	5898	5963	6049	6064#								
NEWDL	71B2	1994\$	2343\$	2974	3018	6740#							
NEXTA1	4EB9	2360#	2372										
NEXTA2	4EDC	2362	2364	2366	2374#								
NEXTA3	4EDF	2375#	2419										
NEXTAD	4EA4	2352#	6688										
NICFLG	7083	5875	5935	6046	6075	6208\$	6357	6575#					
NICR	7084	6051	6053	6055	6057	6182	6186	6577#					
NICRM3	7088	6185	6188	6578#									
NICSUB	6953	6183	6211#	6413									
NOCHK	40FF	444#	4208\$	4376\$									
NOCON	58DB	3753	3804#										
NOPBUF	41A7	597#	1498	2825	2827	3136	3190	3195	3429	3433	4287	4293	4296
		4621	4625\$										4299
NOTBYT	5281	2884	2887#										
NOTL	4F65	2438	2441#										
NOTPVG	5289	2881	2891#										
NOTYPE	40D4	413#	3671\$	4176\$	4469	4920\$	4977\$	5040\$	5046\$				
NPREP	5089	2626	2630#										
NPREP1	5145	2717	2722#										
NSADBF	7206	6791#											
NTEST	56BE	2630	2722	3514#	3915	3963							
NTEST1	56D2	3521	3526#										
NXTSEG	42F8	873#	1217										
OCOPY	40BC	381#											
OFLAG	40BB	380#	1794\$	1914\$	4056\$	4424\$	4475\$	4855\$	5175\$				

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)											
OKSV	0005	21#											
OKV15	0007	26#											
OLDRUN	707D	3574\$	5861\$	5960	6554#								
ONE	5CBB	4327#											
ONEDN	534F	3027	3034#	3477									
ONEUP	5323	2995	3002#	3472	3475								
OPCODE	408F	343#											
OUTADR	7227	2870	6829#										
OUTBUF	7224	2873	6826#										
OUTDAT	7230	2867	2896	6831#									
OUTLET	7225	2860\$	6827#										
OVERRI	7E67	5402	7170#										
PACKS	4172	540#	1028\$	1090									
PAKCNT	40A8	366#	1027\$	1091\$	1114	1121\$	1286\$	1424	1426\$	1448\$			
PARERR	411D	479#											
PARFLG	4197	583#	1526	3159\$	3167\$	3175	3313	3338\$	4069\$	4658	4804\$		
PARITY	5E56	1842	4609#										
PARMSK	007F	222#											
PARSE	4D2B	2070#	2597	2689	3699	3870	3901	3953	4119				
PARSE1	4D33	2073#	2121	2161	2176	2275	2290	2332	2344	2382	2416	2484	
PARSE2	4D50	2078	2084#										
PARSE3	4D53	2083	2085#										
PARSE4	4D71	2098	2108#										
PARSES	4D83	2107	2118#										
PARTMP	4198	584#	3337	4728									
PARVEC	4C22	479	1842#										
PCADR	741E	3786	6917#										
PCMSG	7418	3789	3823	6916#									
PCOUNT	70FC	5178\$	5235	5241\$	5289\$	6645#							
PFFLAG	40B5	372#	3768	4547\$	4575\$	5750							
PHYS	4E6C	2318#	6680										
PHYVIR	52E8	2930	2950	2966#									
PLUS	4F91	2466#	6709										
PMBOOT	6ADA	6464#	6633										
PMHAL0	6647	5755	5760#										
PMHALT	6625	5240	5743#	6609									
PMTALK	6441	5382	5427#										
PMTLK0	6458	5430	5439#										
PNTR1	41A2	594#	1509\$	1514	3430\$								
PNTR2	746C	3345\$	3351	3355\$	6949#								
POINTR	4176	544#	1540\$	1544	1551	2666\$							
POS1	7186	3002	4665	4997	5005	5019	6719#						
POS2	718A	2999	6721#										
POS200	7192	4923	5015	6725#									
POS4	718E	2996	5070	6723#									
PRGCNT	4F79	2433	2450#										

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)												
PRIADR	40C1	390#												
PRICNT	40C3	391#												
PROMPT	4203	631#	823	1916	1929	4361								
PRPSUB	4CCF	1915	1928	1982#										
PSLADR	0000	191#	2441											
PSW	Reserved	4553\$	4609	4637\$	4643\$	6339	6387\$							
PUBCHK	5F8B	4785#	4898											
PUBFLG	40B6	374#	3673\$	3688\$	4785	4803\$	6474\$							
PUBMSG	7323	4788	6873#											
PUPFIL	7427	6920#												
PUVEC	4151	517#	4580											
PVECTR	4C2B	722	862	1845#										
PWRFIL	7239	4816	6835#											
PWRMSG	7341	4592	6877#											
PWRVE1	004C	301#												
PWRVEC	001B	296#	517											
PXXX	4F4B	2429#	6714											
QPEND	40D8	422#	3578\$	5247	5790									
QTOWR	7486	3280	6977#											
QUAL	4115	474#	972	1037	1043	1060	1072	1087	1092	1781\$	1989\$	3849\$	3994	
QUAL1	7218	1988\$	6808#											
QUO	4160	526#	998	1003\$	1335	1339	1347\$	1357	1359\$					
RAD50	482B	1159	1312#											
RAD50A	483A	1313	1315	1317#										
RADCVT	4895	1336	1340	1342	1363#									
RAMADR	71AA	4213	6737#											
RAMCNT	71AE	4216	6738#											
RAMVAL	4080	328#	782	1792\$										
RCVADR	71C2	4093	4102	4950	6753#									
RCVBUF	71C0	2756	6751#											
RCVCSR	7072	3573\$	5317	5579	5639	6542#								
RCVDAT	71C6	1064	2798	2862	4164	4197	4938	4943	4961	5087	6754#			
RCVDB	7073	5301	5773	6543#										
RCVDON	0002	172#												
RCVMOD	71C1	4096	6752#											
RCVMS1	518E	2760#	2764	4182										
RCVMS2	519B	2762	2766#											
RCVMS3	51A3	2771#	2775											
RCVMS4	51B0	2773	2777#											
RCVMS5	518C	2759#	2782											
RCVMSG	5189	2757#	3767											
RCVMSX	5186	1063	1119	2640	2756#	2796	3555	3600	3727	4091	4100	4162	4196	5079
RCVTTR	637B	5317#												
RDCON	7E10	5406	5456	5537	7160#									
RDCON1	7E11	7161#												
RDTEMP	7E12	7163#												
RDWRZ	55A4	3266	3323	3345#										
RDWRZ'	55AC	3348#	3360											
READ	0080	120#	2766	2777	3352	5599	5646	5657	5677	5840	6211	6215	6219	6223
READJ2	0004	18#												

[illegible]

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)									
SNDFLG	40F2	434#									
SAJMS0	5150	2727#									
SNDMS1	5157	2730#	2753	3889							
SNDMS2	515F	2735#	2739								
SNDMS3	516C	2737	2741#								
SNDMS4	5174	2746#	2750								
SNDMS5	5181	2748	2752#								
SNDMSG	514E	1039	1123	2726#	3554	3599	3726	3758	4134	5084	
SNDMSX	514B	2638	2725#	2794	5078						
SNDPAK	408D	341#	1465								
SNDPR1	6214	5066	5069#								
SNDPRP	6204	4929	4933	4940	4947	4958	5063#				
SNDRCV	6222	4886	4992	4999	5007	5017	5024	5030	5069	5077#	
SNDRCX	622B	5037	5081#								
SNDTA	6468	5396	5448#								
SNDTL	6494	5379	5475#								
SNDTL0	649C	5394	5411	5445	5479#						
SNDTL1	64A4	5425	5484#								
SNDTL2	64A6	5454	5472	5485#							
SNDTL3	64A7	5461	5486#								
SNDTR	6475	5401	5404	5444	5456#						
SNDTR1	6483	5458	5462#								
SNDTRL	641E	5398	5406#	5441							
SNDTT0	63E0	5364	5370	5377#							
SNDTT1	63ED	5383#									
SNDTT2	63FD	5386	5391#								
SNDTTX	63C4	5361#									
SNTRL1	642C	5408	5412#								
SOBIAS	40D2	409#									
SOFDON	6AD7	6461#	6632								
SOURCE	7478	3284\$	3306\$	3390	6957#						
SP	Reserved	954	1906\$	4499	4906\$						
SPACE	0020	174#	2040	2054	2395	6702	6828	6830			
SPADRS	749A	7002#									
SPBUFF	4086	333#									
SPCF1	4E4F	2287#									
SPCFLG	0001	180#	2288								
SPCFLW	4E4C	2285#	6705								
SPEND	40D7	421#	3577\$	5205	5243	5291	5543	5574	5591	5785	
SPPACK	7498	3553	4883	7001#							
SPSTRP	4D23	1942	2053#	2057	2560	2580	3633	3738	3818	4030	4052
SRC5IL	4BC0	1813#									
STACK	4080	282#	1906	4906							
STAL10	5760	3584	3611#	6456							
STALL5	5763	3611	3612#								
START	4100	459#									
START1	5815	3702	3715#								
START2	582D	3722	3724#								
STEP	59D6	3953#	6663								
STEP1	00F1	274#	5911								

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)													
STEP2	00F2	276#	5914												
STPA	59EA	3961#	3964												
STPB	59F8	3959	3967#												
STPB1	59FD	3969#	3970												
STPC	59F5	3966#	3974												
STKFLW	4F7E	2453#	6713												
STPFLG	721E	1992\$	3751	3755	3793	3954	6815#								
STRBLK	42FE	876#	1152	1261											
STRPAK	7436	3725	6923#												
STRTA1	4F06	2392#	2406												
STRTA2	4F2E	2394	2408#												
STRTA3	4F45	2396	2398	2400	2418#										
STRTAD	4EF1	2384#	6687												
SUBDIV	4875	971	1334	1338	1346#										
SUBDV1	487B	1348#	1360												
SUBDV2	4888	1352	1356#												
SUCCE\$	4116	475#	701	729	737	1935	2584	3835	4466\$	4478	4481	4497\$	4896	5072	
SUMREG	0020	36#	4560	5232	5920										
SUPRES	526C	2872	2879#												
SUPRS4	5284	2889#	2912												
SUPRS6	527C	2885#	2893												
SWAP	41AA	598#	979\$	981	983	1109\$	1110	2665	2836\$	2838	3189\$	3318	3330	3933	
		3935	4721	4754\$	4755										
SWITCH	4092	346#													
SWPOS	4081	329#	5259												
SYNERR	5D1E	1948	1961	1966	2037	2042	2096	2109	2141	2151	2166	2193	2198	2200	
		2273	2309	2354	2357	2377	2386	2389	2411	2435	2456	2502	2505	2564	
		2568	2573	2578	2582	2600	2692	3184	3632	3648	3653	3656	3740	3820	
		3844	3847	3997	4041	4050	4054	4125	4132	4416#					
SYNMSG	7352	4420	6880#												
SYN_CO*	0003	200#	4419												
S_COMM*	57F2	3698#	6665												
TAB	0009	210#													
TABLE	70A2	5610	6601#												
TALKDB	7E14	5433	7166#												
TALKSR	7E13	5427	7165#												
TALK_V*	080F	318#	1632												
TAPER\$	5D2D	2586	3837	4423#											
TEMP	40B4	370#	1385\$	1386	1666\$	1677	1683\$	1698	1707	1710	4758\$	4773	4775\$		
TEMP1	416A	538#	3194\$	3196\$	3199										
TEST	5A3E	4024#	6666												
THREE	5CC5	4337#													
TIKTOK	7E0D	7155#													
TIMCNT	7051	6522#													
TIMER1	7E0A	7153#													
TIMER2	7E0C	7154#													
TIMSAV	7E08	7152#													
TLKFLG	7E04	1640	5380	5439	5533	7144#									
TMPBUF	4168	537#	1018\$	1024	1317\$	1377	1380\$	3451	3453	3569\$	3590	3937\$	3939	4286	
		4306	4644\$	4646											

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)						
TMRFIL	7052	6313	6524#					
TMRPRO	69BF	6286	6289#					
TMRPR1	69D7	5889	5916	6300#				
TMRPRP	6999	4811	6272#					
TMRSR0	6A28	6347	6349#					
TMRSR1	6A4B	6360	6367#					
TMRSR2	6A55	6370	6372#					
TMRSR3	6A6A	6386#	6398					
TMRSR4	6A6E	6376	6391#					
TMRSRV	6A18	1848	6339#					
TMRVEC	4C2F	1848#						
TOCNT	4083	331#						
TOCNT1	4085	332#						
TODBFR	709A	6412	6414	6589#				
TODPRP	6AB5	6418	6422	6434#				
TODR	6A7D	6404#	6627					
TODW	6A8D	6412#	6628					
TODW1	6A93	6414#						
TPINI1	0058	305#	935	1146	1432			
TPINIT	0055	304#	3566					
TPSUB	69DE	6289	6306#					
TPSUB0	69F2	6315#	6322					
TPSUB1	69F4	6316#	6320					
TRCR	0043	97#						
TRDB	0040	93#	5453\$					
TRKUP1	41B2	601#						
TRKUPC	41B0	600#	980\$	1080	1108\$	1585	1603	1605\$
TRMODE	0042	95#						
TRNFLG	7E06	5208	7148#					
TRSR	0041	94#	5448					
TRYCHK	4924	1419	1423	1432#				
TTCR	004B	114#	5279\$					
TTDB	0048	109#	4590	5273	5484\$			
TTMODE	004A	111#						
TTRCSR	6595	5639#	6603					
TTRDB	6656	5770#	6610					
TTRDB1	6679	5788	5790#					
TTRID	6606	5721#	6608					
TTRIE	65C2	5671#	6606					
TTRRE1	6514	5553#						
TTRREM	64F2	5252	5533#					
TTRTS0	6328	5261	5269#					
TTRTS1	632A	5270#						
TTRTS2	633D	5268	5277	5280#	5564			
TTRTS4	6355	5283	5286	5291#				
TTRTS5	6372	5307	5309#					
TTRTSA	6319	5255	5258	5262#				
TTRTSB	62FE	5246	5250#					
TTRTST	62ED	5237	5243#					
TTSR	0049	110#	5269	5413	5479			

22
.
RC
Er

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)														
UPPREP	5305	2968	2971	2989#												
USED	0080	187#														
USTART	583C	3716	3730#													
VERMSG	729D	1832	6853#													
VNORML	00A3	138#	1525\$	1559\$												
VRTUAL	4E71	2320#	6681													
VSHIFT	00A2	137#	1512\$	1546\$												
WAICMD	5AAA	4087#	4090	6667												
WAITH	6971	5776	5811	6140	6146	6166	6175	6214	6222	6231#						
WAITL	6980	5781	5814	6143	6149	6171	6218	6225	6243#							
WARM	41B7	606#	3669\$	3686\$	4878\$	4911	5033\$	6479\$								
WCS	4E8A	2330#	6682													
WCSB0	0008	29#	1104\$	1617\$	4269\$	4394\$										
WCSB1	0009	30#	1106\$	1621\$	4272\$	4397\$										
WCSB2	000A	31#	1596\$	4275\$	4400\$											
WCSCON	7464	3046	6944#													
WCSDE1	50F0	2677	2679#													
WCSDEP	50C3	2622	2662#													
WCSEX1	521F	2823	2836#	3187												
WCSEX2	521D	2831	2833#													
WCSEXM	51FE	2713	2821#													
WCSLOD	4A09	1405	1585#													
WCS_LO	40D5	415#	1831\$	4070\$	4074	4821\$	4831\$									
WEFLAG	7564	4910\$	5054	5058\$	7118#											
WERROR	7569	5059\$	7121#													
WFADR	71F2	4949	6776#													
WFAIL	61E3	4913	4915	4981	4993	5000	5008	5018	5025	5031	5038	5045#				
WFAIL1	61F2	4930	4932	4934	4936	4941	4948	4959	4971	5053#						
WFATAL	7567	5048\$	7120#													
WFDATA	71F6	4978	6777#													
WFMSG	72CF	5049	6861#													
WFPACK	71F0	5027	6775#													
WORD	4E97	2340#	6678													
WRBYT0	4A36	1594	1617#													
WRBYT1	4A3E	1595	1621#													
WRCRR	7483	3349	6973#													
WRDSUM	409E	357#	1392\$	1398\$	1417	1421										
WRITE	00CC	165#	2733\$	2743\$	4140\$	4150\$	5618\$	5634\$	5775\$	5780\$	5810\$	5813\$	6139\$	6142\$		
		6145\$	6148\$	6152\$	6165\$	6170\$	6174\$									
WRKMSK	40BD	384#														
WRMCNT	71FA	4953	4963	6779#												
WRMSG	72AD	4907	6857#													
WRMST1	609A	4921#	5056	5060												
WRMST2	6104	4958#	4965													
WRMST9	6122	4969#	4975													
WRMSTR	6074	3802	306#													
WRNLSN	747D	3233	4698	6965#												
WRNWRN	7480	3387	6969#													
WRROT8	5633	3348	3418#													
WRRT8X	5638	3420#	3426													

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)												
WRTNOP	4993	977	1497#	2669	2840	3134	3197	3219	3588	3705	4262	4283	4292	4619
		4720	4752											
WRT0Q	7489	3277	6981#											
WRTWCS	5D05	2671	3200	4393#										
WRTWR1	5SE1	3370#	3383											
WRTWR2	5SEE	3373	3375#											
WRTWRZ	5SCA	3231	3276	3294	3327	3362#								
WRZBUF	746E	3300	3322	3326	6950#									
XADRES	40C4	393#												
XCOUNT	40C6	394#												
XCSADR	7078	3580#	5215#	5344	5372#	5632	6550#							
XDBADR	707A	3582#	5217#	5375#	5648	6551#								
XFER	5AD8	4112#	6668											
XFER1	5ADD	4114#	4115											
XFRADR	418A	573#	1033	1096	4120	4224	4230#	4244	4251#	4263				
XFRBUF	4192	576#	3882	4180	4192	4193#	4249#	4294#	4297#	4300#	4357			
XFRCHK	623C	4124	4242	4302	5089#									
XFRCNT	418E	574#	1022#	1077	4316	4354	4365	5089						
XFRCON	5CE7	4135	4220	4265	4361#									
XFRPAK	4188	571#	1036	4133										
XFERRA1	5BDA	4215	4220#											
XFERRAM	5BC2	4128	4212#											
XFRSE0	682B	6050#	6364											
XFRSET	6823	5877	5941	6027	6046#									
XFRWCS	5C27	4130	4261#											
XMEMH	5B28	4143#	4147											
XMEMH1	5B35	4145	4149#											
XMEM1	5B1C	4138#	4155											
XMEM11	5B48	4148	4158#	4159										
XMEM12	5B51	4139	4162#											
XMEM13	5B61	4163	4168#											
XMEML	5B3E	4153#	4157											
XMEMO	5B7B	4136	4180#	4195										
XMEMO1	5BA6	4191	4196#											
XMEMO2	5B94	4183	4190#											
XMEMO4	5B86	4184#	4185											
XMTCSR	7076	3579	3594#	5211	5387	5475	5753	5759#	6547#					
XMTDB	7077	5756	6548#											
XOFIN	5BB7	4200	4203#	4243										
XRAM11	5BE0	4222#	4231											
XRAM12	5BF0	4229#	4237											
XRAM13	5BF7	4227	4233#											
XRAM14	5BFC	4234#	4236#											
XRAMO	5C01	4221	4242#	4253										
XRAMO1	5C11	4249#	4258											
XRAMO2	5C1E	4247	4255#											
XRAMO3	5C22	4256#	4257#											
XVEC	000B	294#	4219											
XWCS1	5C38	4267#	4280											
XWCS11	5C5B	4268	4271	4274	4282#									

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)							
XMCS0	5C64	4266	4286\$						
XMCS01	5C6F	4290\$	4303						
XMCS02	5C9B	4306\$							
XMCS03	5CA1	4284	4308\$						
X_CMND*	73A7	4440	6896\$						
X_DATA*	73C6	4445	6900\$						
YBUSRD	00EC	167\$							
ZBUF1	6AFC	2072	3857	6488\$					
ZBUF2	6AFD	6489\$							
ZERO	7182	5090	6717\$						
ZEROS	70FD	6421	6647\$						
ZROBUF	6AFA	2359	2391	2699	3859	3861	4683	4946	5002
ZROCHK	623F	5090\$							6487\$

=====

=====

Error Addr	Code	Seq	Source statement
		1	
		2	;
		3	ROM011 V10
		4	;
		5	
		6	;
		7	REVISION HISTORY AT END OF FILE!!!!
		8	;
		9	;
=0000		10	UPC14A EQU 00H ;(RO) UPC bit <14>. Asserted high. <MSB>
		11	
=0001		12	MISC2 EQU 01H ;(RO) Misc bit 2 <MSB>
=0001		13	BOOTF EQU 01H ;(RO) Boot Flag. Asserted low. <LSB>
		14	
=0002		15	DCLO EQU 02H ;(RO) DC LO Flag. Asserted high. <MSB>
=0002		16	HLTFLG EQU 02H ;(RO) Halt Flag. Asserted low. <LSB>
		17	
=0003		18	SECURE EQU 03H ;(RO) Disable flag. Asserted low. <LSB>
		19	
=0004		20	READJ2 EQU 04H ;(RO) Bit 07 equals J2. <MSB>
=0004		21	APTFLG EQU 04H ;(RO) APT Present Flag. Asserted low. <LSB>
		22	
=0005		23	OKSV EQU 05H ;(RO) 5 volts OK Flag. Asserted high. <MSB>
		24	
=0006		25	SLWCLK EQU 06H ;(RO) Slow (100 Hz) Clock. <MSB>
=0006		26	REMOTE EQU 06H ;(RO) Remote Flag. Asserted low. <LSB>
		27	
=0007		28	OK15V EQU 07H ;(RO) 15 Volts OK Flag. Asserted high. <MSB>
=0007		29	BOOTEN EQU 07H ;(RO) Boot Enable Flag. Asserted low. <LSB>
		30	
=0008		31	WCSREG EQU 08H ;(WO) WCS Write Data Register
		32	
=001C		33	ITDB EQU 01CH ;(R/W) Timer Data Register
=001D		34	ITCSR EQU 01DH ;(R/W) Timer Status Register
		35	
=0020		36	SUMREG EQU 020H ;(RO) Ready Bit Summary Reg.
		37	
		38	; All bits asserted low.
		39	
		40	; 07 06 05 04 03 02 01 00
		41	;Timer Power Remote TU-58 Local TU-58 Remote Local
		42	;Int. Fail RCVR RCV RCV XMIT XMIT XMIT
		43	; Time Done Done Done Ready Ready Ready
		44	; Out
		45	
=0020		46	CLRCLK EQU 020H ;(WO) Clear CPU Clock Run
=0021		47	SETCLK EQU 021H ;(WO) Set CPU Clock Run
		48	
=0022		49	CLRSS EQU 022H ;(WO) Clear Clock Single Step
=0023		50	SETSS EQU 023H ;(WO) Set Clock Single Step

Error Addr	Code	Seq	Source statement
		51	
=0024		52	CLRCSK EQU 024H ;(W0) Clear CSR clock
=0025		53	SETCSK EQU 025H ;(W0) Set CSR clock
		54	
=0026		55	CLRUPK EQU 026H ;(W0) Clear the UPC clock
=0027		56	SETUPK EQU 027H ;(W0) Set the UPC clock
		57	
=0028		58	SETMCI EQU 028H ;(W0) Set Memory Control Init
=0029		59	CLRMCI EQU 029H ;(W0) Clear Memory Control Init
		60	
=002A		61	SETMCK EQU 02AH ;(W0) Set Memory Control Clock
=002B		62	CLRMCK EQU 02BH ;(W0) Clear Memory Control Clock
		63	
=002C		64	CLRTMR EQU 02CH ;(W0) Stop timer
=002D		65	SETTMR EQU 02DH ;(W0) Start timer counting
		66	
=002E		67	CLRBSY EQU 02EH ;(W0) Clear Unibus Bus Busy
=002F		68	SETBSY EQU 02FH ;(W0) Set Unibus Bus Busy
		69	
=0030		70	SETDCL EQU 030H ;(W0) Set DC LO
=0031		71	CLRDCL EQU 031H ;(W0) Clear DC LO
		72	
=0032		73	SETACL EQU 032H ;(W0) Set AC LO
=0033		74	CLRACL EQU 033H ;(W0) Clear AC LO
		75	
=0034		76	INITL EQU 034H ;(W0) Clear Unibus Init
=0035		77	INITH EQU 035H ;(W0) Set Unibus Init
		78	
=0036		79	CLRWCK EQU 036H ;(W0) Clear WCS Write Clock.
=0037		80	SETWCK EQU 037H ;(W0) Set WCS Write Clock.
		81	
=0038		82	CLRCSR EQU 038H ;(W0) Clear the CSR Serial Input.
=0039		83	SETCSR EQU 039H ;(W0) Set the CSR Serial Input.
		84	
=003A		85	CLRBLK EQU 03AH ;(W0) Clear the 100Hz Block.
=003B		86	SETBLK EQU 03BH ;(W0) Set the 100Hz Block.
		87	
=003C		88	CLRLIT EQU 03CH ;(W0) Clear the Run Light
=003D		89	SETLIT EQU 03DH ;(W0) Set the Run Light
		90	
=003E		91	CLRRD EQU 03EH ;(W0) Clear R/D light
=003F		92	SETRD EQU 03FH ;(W0) Set R/D light
		93	
=0040		94	TRDB EQU 040H ;
=0041		95	TRSR EQU 041H ;
=0042		96	TRMODE EQU 042H ;
=0043		97	TRCR EQU 043H ;
		98	
		99	
=0044		100	TUDB EQU 044H ;(R/W) TU-58 Data Buffer

Error	Addr	Code	Seq	Source	statement
		=0045	101	TUSR	EQU 045H ;(R0) TU-58 Status Register
		=0046	102	TUMODE	EQU 046H ;(R/W) TU-58 Mode Registers 1 and 2. First access
			103		;gets MR1, second gets MR2. Reading the TUCR
			104		;forces the internal pointer back to MR1.
		=0047	105	TUCR	EQU 047H ;(R/W) TU-58 Command Register
			106		
			107		
		=0048	108	TTDB	EQU 048H ;(R/W) Terminal Data Buffer
		=0049	109	TTSR	EQU 049H ;(R0) Terminal Status Register
		=004A	110	TTMODE	EQU 04AH ;(R/W) Terminal Mode Register 1 and 2. First access
			111		;gets MR1, second gets MR2. Reading the CR forces
			112		;the internal pointer back to MR1
		=004B	113	TTCR	EQU 04BH ;(R/W) Terminal Command Register
			114		
			115		
			116		
			117		
			118		
		=0080	119	READ	EQU 080H ;(R0) Console Read Register
			120		
		=0082	121	CPUACK	EQU 082H ;(R0) Acknowledge from CPU. <LSB>
			122		
		=0083	123	CPATTN	EQU 083H ;(R0) Attention from CPU. <LSB>
			124		
		=0084	125	UPC14	EQ ' 084H ;(R0) UPC bit <14>. <LSB>
			126		
		=0085	127	CSR07	EQU 085H ;(R0) CSR bit <07>. <LSB>
			128		
		=0086	129	CSR15	EQU 086H ;(R0) CSR bit <15>. <LSB>
			130		
		=0087	131	CSR23	EQU 087H ;(R0) CSR bit <23>. <LSB>
			132		
		=00A0	133	ENBPAR	EQU 0A0H ;(W0) Enable Stall on Parity Error
		=00A1	134	DISPAR	EQU 0A1H ;(W0) Disable Stall on Parity Error
			135		
		=00A2	136	VSHIFT	EQU 0A2H ;(W0) Set the V-BUS to Shift Mode
		=00A3	137	VNORML	EQU 0A3H ;(W0) Set the V-BUS to Normal Mode
			138		
		=00A4	139	SETTIM	EQU 0A4H ;(W0) Set the Interval Timer Interrupt
		=00A5	140	CLR*IM	EQU 0A5H ;(W0) Clear the Interval Timer Interrupt
			141		
		=00A6	142	ENBMEM	EQU 0A6H ;(W0) Enable Memory References
		=00A7	143	DISMEM	EQU 0A7H ;(W0) Disable Memory References
			144		
		=00A8	145	SETACK	EQU 0A8H ;(W0) Set Console Acknowledge
		=00A9	146	CLRACK	EQU 0A9H ;(W0) Clear Console Acknowledge
			147		
			148		;NOTE: The Console Acknowledge signal is also used as the UPC
			149		;Serial Input. When used this way the sense is inverted. To simplify
			150		;matters the next two equates are included.

Error	Addr	Code	Seq	Source statement
-------	------	------	-----	------------------

			151	
=00A8			152	CLRUPC EQU 0A8H ;(W0) Clear the V-BUS Serial Input
=00A9			153	SETUPC EQU 0A9H ;(W0) Set the V-BUS Serial Input
			154	
=00AA			155	SETATN EQU 0AAH ;(W0) Set Console Attention
=00AB			156	CLRATN EQU 0ABH ;(W0) Clear Console Attention
			157	
=00AC			158	SETPFI EQU 0ACH ;(W0) Set Power Fail Interrupt
=00AD			159	CLRPFI EQU 0ADH ;(W0) Clear Power Fail Interrupt
			160	
=00AE			161	SETHLT EQU 0AEH ;(W0) Set the Console Halt bit
=00AF			162	CLRHLT EQU 0AFH ;(W0) Clear the Console Halt bit
			163	
=00CC			164	WRITE EQU 0CCH ;(W0) The Console Write Register
			165	
=00EC			166	YBUSRD EQU 0ECH ;(R0) Y-BUS Read
			167	
			168	
			169	

;The following are some definitions

			170		
			171		
=4080			172	STACK EQU 4080H ;Beginning address of Stack Pointer	
=1E62			173	DELAY EQU 7778 ;70 msec delay for Power Fail	
=0004			174	TUINIT EQU 4 ;TU Init value	
=000D			175	BREAK EQU 0DH ;Break value	
=000D			176	CR EQU 0DH ;	
=000A			177	LF EQU 0AH ;	
=000A			178	LINE_FEED EQU 0AH ;	
=000D			179	CAR_RET EQU 0DH ;	
=0003			180	CNTRLC EQU 3 ;	
=0010			181	CNTRLP EQU 10H ;ASCII for Control P	
=0013			182	CNTRLS EQU 13H ;	
=0011			183	CNTRLQ EQU 11H ;	
=000F			184	CNTRLO EQU 0FH ;	
=0002			185	RCVDON EQU 02 ;	
=000B			186	SIMMSK EQU 0BH ;	
=0035			187	TTCR_CODE EQU 35H ;	
			188		
			189		
=4080			190	RAM_VALID EQU 04080H ;RAM valid flag byte	
			191		
=4081			192	SWITCH_POS EQU 4081H ;Address of Switch Position Flag	
			193		
=4082			194	BYTSUM EQU 04082H ;Address of byte sum area	
			195		
=4083			196	TOCNT EQU 04083H ;This is the address of the byte wide Time Out	
=4084			197	TOCNT1 EQU 04084H ;This is the address of the word wide Time Out	
			198		
=4086			199	SPBUFF EQU 04086H ;Address of the SP Buffer	
=4088			200	HLBUFF EQU 04088H ;Address of HL buffer	

Error Addr	Code	Seq	Source statement
		201	
=408A		202	COLD EQU 0408AH ;Cold flag
=408B		203	CODE_FLAG EQU 0408BH ;Code flag (0=Console, 1=Micmon).
		204	
=408C		205	RETRY EQU 0408CH ;
		206	
=408D		207	SNDPAK EQU 0408DH ;Address of Send Packet
=4091		208	UNIT EQU 04091H ;Address of Unit Number in Send Packet
		209	
=4099		210	MODEM_ADDRESS EQU 4099H ;
		211	
=409B		212	LODFLG EQU 409BH ;
=409C		213	DIRFLG EQU 409CH ;
=409D		214	INDFLG EQU 409DH ;
		215	
=409E		216	WORD_SUM EQU 409EH ;
		217	
=40A0		218	SNDADR EQU 40A0H ;
=40A2		219	SNDCNT EQU 40A2H ;
		220	
=40A3		221	LODADR EQU 40A3H ;
=40A7		222	LODCNT EQU 40A7H ;
		223	
=40A8		224	PAKCNT EQU 40A8H ;
		225	
=40AA		226	ENDPAK EQU 40AAH ;Address of the End Packet area
		227	
=40B4		228	TEMP EQU 40B4H ;
		229	
=40B5		230	PFFLAG EQU 40B5H ;
=40B6		231	PUBFLG EQU 40B6H ;
		232	
=40B7		233	MSGADR EQU 40B7H ;
=40B9		234	MYPC EQU 40B9H ;
		235	
=40BB		236	O_FLAG EQU 40BBH ;
=40BC		237	O_FLAG_COPY EQU 40BCH ;
		238	
=40BD		239	LR_MASK_WORK EQU 40BDH ;
=40BE		240	LR_MASK EQU 40BEH ;
=40BF		241	LR_MASK_COPY EQU 40BFH ;
		242	
=40C0		243	SAVE_CHARACTER EQU 40C0H ;
		244	
=40C1		245	PRIADR EQU 40C1H ;
=40C3		246	PRICNT EQU 40C3H ;
		247	
=40C4		248	XADRES EQU 40C4H ;
=40C6		249	XCOUNT EQU 40C6H ;
		250	

Error Addr	Code	Seq	Source statement
=40C8		251	CNTADR EQU 40C8H ;
=40CA		252	BUFADR EQU 40CAH ;
		253	
=40CC		254	RETRYI EQU 40CCH ;
=40CD		255	S_FLAG EQU 40CDH ;
		256	
=40CE		257	DELETE_FLAG EQU 40CEH ;
		258	
		259	
=40D0		260	SILO_ACTIVE EQU 40D0H ;Silo Active Flag
=40D1		261	SILO_IN_INDX EQU 40D1H ;(Byte) Silo Input Index
=40D2		262	SILO_OUT_INDX EQU 40D2H ;(Byte) Silo Output Index
		263	
=40D3		264	RUN_FLAG EQU 40D3H ;Run Flag (indicates Program Mode)
		265	
=40D4		266	NOTYPE EQU 40D4H ;
		267	
=40D5		268	WCS_LOAD_FLAGS EQU 40D5H ;
		269	
=40D6		270	APTLOD EQU 40D6H ;
		271	
=40D7		272	SPEND EQU 40D7H ;
=40D8		273	QPEND EQU 40D8H ;
		274	
=40D9		275	MIPFLG EQU 40D9H ;
		276	
=40DA		277	DISCNT EQU 40DAH ;Disconnect Count
		278	
=40F0		279	MSGBUF EQU 40F0H ;
		280	
=40F2		281	SEND_IN_PROGRES EQU 40F2H ;
		282	
=40F3		283	CPFLAG EQU 40F3H ;
		284	
=40F4		285	SECURE_STATUS EQU 40F4H ;(B) Keyswitch Secure Position status.
		286	
=40FE		287	CTL_CHR_DISABLE EQU 40FEH ;(B) Control C and P disable flag
=40FF		288	NOCHK EQU 40FFH ;(B) CR check for auto LF disable
		289	
		290	
		291	
=4100		292	START EQU 4100H ;
=4103		293	GOSV EQU 4103H ;
		294	
=4116		295	SUCCESS EQU 4116H ;This is the address of the Success Code byte
		296	
=411D		297	PARERR EQU 411DH ;Address of Parity Error handler
=4120		298	ACLO EQU 4120H ;Address of ACLO handler
		299	
=414E		300	OVECTOR EQU 414EH ;

Error	Addr	Code	Seq	Source statement
			301	
	=4B80		302	CHAR_SILO EQU 4B80H ;64 byte character silo
	=4BC0		303	SOURCE_SILO EQU 4BC0H ;64 byte source flag silo
			304	
	=4C28		305	CVECTR EQU 4C28H ;Control C vector
	=4C2B		306	PVECTR EQU 4C2BH ;Control P vector
			307	
	=4C2F		308	TMRVEC EQU 4C2FH ;Interval Timer interrupt vector
			309	
	=7000		310	CHRCNT EQU 7000H ;
	=7001		311	CHRBUF EQU 7001H ;
			312	
			313	;.....
			314	;
	=7E00		315	REMRSR EQU 7E00H ;(B)
	=7E01		316	REMRDB EQU 7E01H ;(B)
			317	
	=7E02		318	REMXSR EQU 7E02H ;(B)
	=7E03		319	REMXDB EQU 7E03H ;(B)
			320	
	=7E04		321	TALK_FLAG EQU 7E04H ;(B)
			322	
	=7E05		323	DISCARD_RO EQU 7E05H ;(B) Discard Remote Output flag.
			324	
	=7E06		325	TRNFLG EQU 7E06H ;(B)
			326	
	=7E07		327	MSG_TO_SEND EQU 7E07H ;(B)
			328	
	=7E08		329	TIMSAV EQU 7E08H ;(W)
	=7E0A		330	TIMER1 EQU 7E0AH ;(W)
	=7E0C		331	TIMER2 EQU 7E0CH ;(B)
	=7E0D		332	TIKTOK EQU 7E0DH ;(B)
			333	
	=7E0E		334	LOGCON EQU 7E0EH ;(B)
	=7E0F		335	LOGCNT EQU 7E0FH ;(B)
			336	
	=7E10		337	RDCON EQU 7E10H ;(B)
	=7E11		338	RDCON1 EQU 7E11H ;(B)
			339	
	=7E12		340	RDTEMP EQU 7E12H ;(B)
			341	
	=7E13		342	TALKSR EQU 7E13H ;(B)
	=7E14		343	TALKDB EQU 7E14H ;(B)
			344	
			345	
	=7E16		346	ICNT EQU 7E16H ;(W)
	=7E18		347	IADDR EQU 7E18H ;(W)
			348	
	=7F1A		349	DPFLAG EQU 7E1AH ;(B)
	=7E1B		350	INLOCK EQU 7E1BH ;(B)

Error Addr	Code	Seq	Source statement
=7E1C		351	HEADER EQU 7E1CH ;(B)
=7E1D		352	MESSAG EQU 7E1DH ;(B)
		353	
=7E1E		354	INFLAG EQU 7E1EH ;(B)
		355	
=7E1F		356	MSG_TO_READ EQU 7E1FH ;(B)
		357	
=7E20		358	SOHFLG EQU 7E20H ;(B)
		359	
=7E21		360	ACTCNT EQU 7E21H ;(W)
=7E23		361	ACTBUF EQU 7E23H ;(W)
		362	
=7E25		363	LOCCNT EQU 7E25H ;(W)
		364	
=7E27		365	OUTCNT EQU 7E27H ;(B) Output Data count
=7E28		366	OUTBUF EQU 7E28H ;(W) Output Data Buffer address
		367	
=7E2A		368	LAST_GOOD_MSG EQU 7E2AH ;(B) Last Good Message received
=7E2B		369	LAST_TYPE_SENT EQU 7E2BH ;(B) Last Message Sent Type flag
		370	
=7E2C		371	RD_LD_FLAG EQU 7E2CH ;(B) RD Load Message Type in Progress.
=7E2D		372	RD_LD_ERR EQU 7E2DH ;(B) RD Load Error flag.
		373	
=7E30		374	RCRC EQU 7E30H ;Receive CRC
=7E32		375	XCRC EQU 7E32H ;Xmit CRC
		376	
=7E34		377	CRCCN0 EQU 7E34H ;(B)
=7E35		378	CRCCN1 EQU 7E35H ;(B)
		379	
=7E36		380	NEEDI EQU 7E36H ;(B)
		381	
=7E37		382	ASC_MSG_PTR EQU 7E37H ;(W)
=7E39		383	ASC_MSG_COUNT EQU 7E39H ;(B)
		384	
=7E3A		385	ENTER_TRANS EQU 7E3AH ;(B)
		386	
=7E3B		387	RDSPER EQU 7E3BH ;(B)
		388	
=7E40		389	IBUF_CNT_COPY EQU 7E40H ;(W)
=7E42		390	IBUF_ADR_COPY EQU 7E42H ;(W) Input Buffer Address Copy.
		391	
=7E44		392	MSG15_PTR EQU 7E44H ;(W) Contains address of Msg Type 15 String.
=7E46		393	MSG15_COUNT EQU 7E46H ;(B) Count for Msg Type 15 string.
		394	
=7E47		395	CLMSG_PTR EQU 7E47H ;(W) Pointer into Connection Lost Msg.
=7E49		396	CLMSG_COUNT EQU 7E49H ;(B) Count for Connection Lost Message.
		397	
=7E4A		398	RD_LD_CPTR EQU 7E4AH ;(W)
=7E4C		399	RD_LD_CCOUNT EQU 7E4CH ;(B)
=7E4D		400	RD_LD_DPTR EQU 7E4DH ;(W)

Error	Addr	Code	Seq	Source statement
	=7E4F		401	RD_LD_DCOUNT EQU 7E4FH ;(B)
	=7E50		402	RD_LD_EPTR EQU 7E50H ;(W)
	=7E52		403	RD_LD_ECOUNT EQU 7E52H ;(B)
	=7E53		404	RD_LD_OE EQU 7E53H ;(B)
	=7E54		405	RD_LD_SUCCESS EQU 7E54H ;(B)
	=7E55		406	RD_LD_CBUFFER EQU 7E55H ;(10B)
			407	
			408	
	=7E60		409	IBUF1 EQU 7E60H ;(B) One byte input buffer
			410	
			411	;
			412	;
	=7E61		413	CTL_PTR EQU 7E61H ;(W)
	=7E63		414	CTL_COUNT EQU 7E63H ;(B)
	=7E64		415	CTL_BUFFER EQU 7E64H ;(4B) The first byte is used to hold the
			416	;count of how many control bytes. The second
			417	;the information about Talk, Talk Echo, Discard
			418	;Remote, Parallel Control and Local Copy. The
			419	;third the status of Keyswitch Secure. The
			420	;fourth (OVERRIDE_LCS) the Override Local Copy
			421	;in Secure.
	=7E67		422	OVERRIDE_LCS EQU 7E67H ;Last byte of CTL_BUFFER
			423	;
			424	;
			425	
			426	
			427	;
			428	;
	=7E68		429	HBUFR EQU 7E68H ;Header Buffer 1
			430	
	=7E69		431	HCOUNT EQU 7E69H ;For SOH type headers (two bytes)
	=7E69		432	CTLTYP EQU 7E69H ;For ENQ type headers
	=7E6A		433	CTLREA EQU 7E6AH ;For ENQ type headers
			434	
	=7E6B		435	LMSGNM EQU 7E6BH ;Last Message Number
	=7E6C		436	CMSGNM EQU 7E6CH ;Current Message Number
	=7E6D		437	STATION EQU 7E6DH ;
	=7E6E		438	HCRC EQU 7E6EH ;
			439	;
			440	;
			441	
	=7E70		442	CTL_C_INTR EQU 7E70H ;
			443	
	=7E71		444	LOCAL_COPY EQU 7E71H ;
	=7E72		445	PARALLEL_CNTRL EQU 7E72H ;
			446	
	=7E73		447	TALK_COPY EQU 7E73H ;
			448	
	=7E76		449	MSGFLG EQU 7E76H ;MSG TYPE to be sent
	=7E77		450	MSGTYP EQU 7E77H ;MSG TYPE

Error	Addr	Code	Seq	Source statement
			451	
	=7E8D		452	LOBUF EQU 7E8DH ;(B) Indicates which output buffer is Locked
	=7E8E		453	OUTLOC EQU 7E8EH ;(B) Indicates there is a Locked Output buffer
	=7E8F		454	LBSIP EQU 7E8FH ;(B) Locked Buffer Send In Progress flag
	=7E90		455	LBSENT EQU 7E90H ;(B) Indicates that the locked buffer has been
			456	;sent
			457	
	=7E91		458	ACTFLG EQU 7E91H ;(B) Indicates which buffer is active
			459	
	=7E94		460	HDCNT EQU 7E94H ;(B)
	=7E95		461	HDADR EQU 7E95H ;(W)
			462	
	=7E97		463	CTLFLG EQU 7E97H ;(B)
			464	
	=7E98		465	SOHENQ EQU 7E98H ;(B)
			466	
	=7E9A		467	BLDBLK EQU 7E9AH ;BLDBLK = Address of Header to be sent
			468	;BLDBLK+2 = CTL flag
			469	;BLDBLK+3 = Output Buffer count
			470	;BLDBLK+4 = Output Buffer address
			471	;BLDBLK+6 = MSGTYP flag
			472	
	=7EA1		473	SOHHED EQU 7EA1H ;
			474	
	=7EA7		475	ENQHED EQU 7EA7H ;
			476	
	=7EDB		477	IBUF0 EQU 7EDBH ;128 byte input buffer (plus 3 bytes).
	=7F5E		478	OCNT0 EQU 7F5EH ;
	=7F5F		479	OBUF0 EQU 7F5FH ;80 Byte Output Buffer.
	=7FAF		480	OCNT1 EQU 7FAFH ;
	=7FB0		481	OBUF1 EQU 7FB0H ;80 Byte Output Buffer.
			482	;
			483	;.....
			484	
			485	
	=0000		486	ORG 0
0000	F3		487	BEGIN: DI ;
0001	31 0000		488	LXI SP,0 ;Zero the flag that determines which code is
			489	;to be loaded
0004	C3 0815		490	JMP ST_VECTOR ;Go see if we should do the self test
			491	
			492	
			493	
	=0008		494	ORG 08H
0008	C3 0274		495	RSTN1: JMP ROM_IDLE_PREP ;Go to the ROM Idle Loop.
			496	
000B	C3 02E9		497	XVEC: JMP X_CMND_0 ;External entry into Rom based X command
			498	
	=0010		499	ORG 010H ;
0010	C3 01D6		500	RSTN2: JMP TU_RCVR_DONE ;Go to software interrupt #2 routine

Error	Addr	Code	Seq	Source statement
-------	------	------	-----	------------------

Address	Hex	Assembly	Comment
0013	C3 0215	MODVEC: JMP	MODEM_CHECK ;External entry into the Modem routines
	=0018		
0018	C3 05D4	RSTN3: JMP	018H ;Go to software interrupt #3 routine
001B	C3 04DF	PMRVEC: JMP	POWER_UP ;External entry to Power Up code
	=0020		
0020	C3 0427	RSTN4: JMP	020H ;Go to software interrupt routine #4
0023	00	NOP	;Filler
0024		INIT_VECTOR:	
0024	C3 0112	JMP	INIT ;Go to INIT.
0027	00	NOP	;Filler
0028	C3 05B9	RSTN5: JMP	SEND_LI_0 ;Jump to software interrupt #5
002b	00	NOP	;Filler
002C	F3	RST55: DI	;Disable further Interrupts
002D	C3 4C2F	JMP	TMRVEC ;Go to the timer routines
0030	C3 05B7	RSTN6: JMP	SEND_LINE ;Go to software interrupt #6 routine
0033	00	NOP	;Filler
0034	F3	RST65: DI	;Disable further Interrupts
0035	C3 411D	JMP	PARERR ;Go to the Parity error code
0038	C3 06A5	RSTN7: JMP	DPRINT ;Go to the Diagnostic Print code
003B	00	NOP	;Filler
003C	F3	RST75: DI	;Disable further Interrupts
003D	C3 04CE	JMP	ACLCHK ;Go to AC LO check

```

532 ;#####
533 ;#####
534 ;
535 ;The following routines must not be moved because they are called from the RAM
536 ;
537 ;    INIVEC
538 ;    SCVECO
539 ;    GS_VECTOR
540 ;    RSVEC
541 ;    PWRVE1
542 ;    FU_VECTOR
543 ;    DUMMY2
544 ;    INIT_DRIVES
545 ;    TUERR2
546 ;    TUERR3
547 ;    TUERR
548 ;    TU_SEND_PACKET
549 ;    CHKSUM_WORD
550 ;    COPY

```

Error	Addr	Code	Seq	Source statement	
			551	;	COPY1
			552	;	SAVESP
			553	;	SAVSP1
			554	;	
0040	C3	0160	555	INIVC: JMP	INIT1 ;Vector for RAM
0043	C3	05DA	556	SCVEC0: JMP	SEND_CHAO ;Vector in Send Character routine
0046			557	GS_VECTOR:	
0046	C3	03F1	558	JMP	GET_SILO ;Go get the silo character
0049	C3	040F	559	RSVEC: JMP	READ_SILO ;Go read the top character from the silo
004C	C3	04EA	560	PWRVE1: JMP	POWER_0 ;Second entry in Power Up routine
004F			561	FU_VECTOR:	
004F	C3	0691	562	JMP	FIXUP ;Go to the Modem Fixup routine.
0052	C3	0052	563	DUMMY1: JMP	DUMMY1 ;<For future use>
			564		
0055			565	INIT_DRIVES:	
0055	CD	0104	566	CALL	SAVESP ;Go zero Success and save the SP
0058	21	40CC	567	LXI	H,RETRYI ;Point to the Tape Init Retry count area
005B	36	09	568	MVI	M,09 ;Set up a retry count
005D	CD	03E0	569	1\$: CALL	BRKSUB ;Go send a break to the TU-58
0060	21	06B6	570	LXI	H,INITPK ;Point to this packet
0063	CD	00D1	571	CALL	TU_SEND_0 ;Go send it
0066	CD	020B	572	CALL	TOPREP ;Go Prep the Time Out counts
0069	CD	01DB	573	CALL	TU_RCVR_0 ;Go wait for TU RCVR DONE
006C	FE	10	574	CPI	10H ;Look for a Continue code from the TU-58
006E	C8		575	RZ	;Leave if it is
006F	21	40CC	576	LXI	H,RETRYI ;Point to the Tape Init Retry count
0072	35		577	DCR	M ;Subtract one
0073	C2	005D	578	JNZ	1\$;Loop back if still positive
0076	CD	0079	579	CALL	TUERR2 ;Error if not
			580	;	
			581	;	
			582		
			583		
			584	;	
			585	;	
0079	3E	02	586	TUERR2: MVI	A,02 ;Error code
007B	21	06CE	587	LXI	H,DEVERR ;Address of Device Error message
007E	C3	0086	588	JMP	TUERR ;Go to some common code
			589		
0081	3E	03	590	TUERR3: MVI	A,3 ;Error code
0083	1	06E1	591	LXI	H,REDERR ;Address of Read Error error message
0086	32	4116	592	TUERR: STA	SUCCESS ;Store the error code
0089	22	40B7	593	SHLD	MSGADR ;Store the Error Message address
008C	E1		594	POP	H ;Get the error PC for the 8085
008D	22	40B9	595	SHLD	MYPC ;Save it
0090	2A	4086	596	LHLD	SPBUFF ;Get the SP value
0093	F9		597	SPHL	;Put it in the SP
0094	C9		598	RET	;Leave
			599	;	
			600	;	

Error	Addr	Code	Seq	Source statement
			601	
			602	;
			603	;
			604	;
			605	;
			606	;
			607	;
			608	;
			609	;
			610	;
			611	;
			612	;
			613	;
			614	;
			615	;
			616	;
			617	;
			618	;
			619	;
0095			620	TU_SEND_PACKET:
0095	3A 40D9		621	LDA MIPFLG ;See if we are trying to connect RD.
0098	1F		622	RAR ;Put the flag in the carry
0099	3E 00		623	MVI A,0 ;Prime A with a 0.
009B	D2 00A0		624	JNC 1\$;IF NOT MODEM IN PROGRESS, THEN jump.
009E	3E 08		625	MVI A,8 ;ELSE, make an 8 to set up Handshake Mode
00A0	32 4092		626	1\$: STA SNDPAK+5 ;Put it in the Switch Byte of SNDPAK
00A3	21 408D		627	LXI H,SNDPAK ;Point to this packet
00A6	3E 0C		628	MVI A,12 ;Set up a count
00A8	32 40A2		629	STA SNDCNT ;and save it
00AB	22 40A0		630	SHLD SNDADR ;Save the address
00AE	21 0000		631	LXI H,0 ;Now let's zero the checksum work area
00B1	22 409E		632	SHLD WORD_SUM ;Do it
00B4	CD 01C8		633	2\$: CALL TU_XMIT_READY ;Look for Xm't Rdy
00B7	2A 40A0		634	LHLD SNDADR ;Get the address of the data
00BA	7E		635	MOV A,H ;Get the data
00BB	D3 44		636	OUT TUDB ;Send the data out
00BD	23		637	INX H ;Bump the pointer
00BE	22 40A0		638	SHLD SNDADR ;Save the new address
00C1	21 40A2		639	LXI H,SNDCNT ;Point to the count
00C4	CD 00DE		640	CALL CHKSUM_WORD ;Go checksum the byte as part of a 16 bit word.
00C7	21 40A2		641	LXI H,SNDCNT ;Point to the count
00CA	35		642	DCR M ;Reduce it by one
00CB	C2 00B4		643	JNZ 2\$;Loop back if not zero
00CE	21 409E		644	LXI H,WORD_SUM ;Point to the Checksum
00D1			645	TU_SEND_0:
00D1	CD 00D4		646	CALL 1\$;Call the next part
00D4	E5		647	1\$: PUSH H ;Save the address
00D5	CD 01C8		648	CALL TU_XMIT_READY ;Wait for Xmit Rdy
00D8	E1		649	POP H ;Get the address back
00D9	7E		650	MOV A,H ;Get the byte it's pointing to

Error	Addr	Code	Seq	Source statement
	00DA	D3 44	651	OUT TUDB ;Send it out
	00DC	23	652	INX H ;Bump the address
	00DD	C9	653	RET ;Leave
			654	;
			655	;
			656	;
			657	;
			658	;
			659	;
			660	;
	00DE		661	CHKSUM_WORD:
	00DE	5F	662	MOV E,A ;Assume low byte for now
	00DF	47	663	MOV B,A ;Save a copy in B
	00E0	7E	664	MOV A,M ;Get the count
	00E1	1F	665	RAR ;Check the LSB
	00E2	DA 00EA	666	JC 1\$;Jump if the count is odd
	00E5	16 00	667	MVI D,0 ;and zero the high byte
	00E7	C3 00ED	668	JMP 2\$;Go add them
	00EA	53	669	1\$: MOV D,E ;Put the character in the high byte
	00EB	1E 00	670	MVI E,0 ;and zero the low byte
	00ED	2A 409E	671	2\$: LHLD WORD_SUM ;Get the current value of the checksum
	00F0	19	672	DAD D ;Double add to HL
	00F1	D2 00F5	673	JNC 3\$;Jump if no carry
	00F4	23	674	INX H ;This takes care of the end around carry
	00F5	22 409E	675	3\$: SHLD WORD_SUM ;Store the value
	00F8	C9	676	RET ;and leave
			677	;
			678	;
			679	;
			680	;
			681	;
			682	;
			683	;
			684	;
			685	;
			686	;
	00F9	06 04	687	COPY: MVI B,04H ;Set up the loop count.
	00FB		688	COPY1:
	00FB	1A	689	1\$: LDAX D ;Get the source byte
	00FC	77	690	MOV M,A ;Write it to the destination.
	00FD	13	691	INX D ;Bump the source pointer.
	00FE	23	692	INX H ;Bump the destination pointer
	00FF	05	693	DCR B ;Decrement the count
	0100	C2 00FB	694	JNZ 1\$;Loop back if not zero
	0103	C9	695	RET ;Else, return
			696	;
			697	;
			698	;
			699	;
			700	;

Error	Addr	Code	Seq	Source statement
			701	:
			702	: SAVESP
			703	:
			704	: This routine is used by routines that could encounter a fatal error.
			705	: Because we want to abort, and because the depth on the Stack may be
			706	: indeterminant, the code that calls such routines calls SAVESP first. Thus,
			707	: when such an error occurs, the error routines can restore the SP to the
			708	: correct level before returning.
			709	:
0104	AF		710	SAVESP: XRA A ;Make a 0
0105	32 4116		711	STA SUCCES ;Zero the Success code byte
0108	EB		712	SAVSP1: XCHG ;Save HL in DE
0109	21 0002		713	LXI H,2 ;Put an offset into HL
010C	39		714	DAD SP ;Add in the SP value
010D	22 4086		715	SHLD SPBUFF ;Save it
0110	EB		716	XCHG ;Get HL back
0111	C9		717	RET ;Leave
			718	:
			719	:
			720	:
			721	: End of fixed routines
			722	:
			723	:
			724	:
			725	:
			726	:
			727	:
			728	:
			729	: INIT
			730	:
			731	: The first part of INIT cleans up the machine.
			732	:
0112	21 4000		733	INIT: LXI H,4000H ;Point to the beginning of RAM
0115	AF		734	1\$: XRA A ;Make a 0
0116	77		735	MOV M,A ;Zero memory
0117	23		736	INX H ;Bump the pointer
0118	7C		737	MOV A,H ;Get the high byte
0119	17		738	RAL ;See if the MSB set
011A	D2 0115		739	JNC 1\$;Loop back if not
011D	21 0000		740	LXI H,0 ;Clear HL
0120	39		741	DAD SP ;Get the Code Flag out of the SP
0121	7D		742	MOV A,L ;It's in the low byte
0122	32 408B		743	STA CODE_FLAG ;Initialize the Code Flag
0125	D3 3B		744	OUT SETBLK ;block the 100HZ clock
0127	3E 17		745	MVI A,17H ;code to force internal pointer to MM reg
0129	D3 1D		746	OUT ITCSR ;send it out
012B	DB 1C		747	IN ITDB ;get the low byte of the MM reg
012D	FE 0C		748	CPI 0CH ; low byte should be 0C if BBU is present
012F	C2 0139		749	JNZ 2\$;if low byte is not correct then jump
0132	DB 1C		750	IN ITDB ;else check high is also correct

Error	Addr	Code	Seq	Source statement
0134	FE 0A	751		CPI 0AH ;this should be high byte
0136	CA 0144	752		JZ 3\$;if everything ok then unblock 100HZ clk
0139	3E 17	753	2\$:	MVI A,17H ;We need this value to force the Internal
013B	D3 1D	754		OUT ITCSR ;Pointer of the 9513 to point at the MM Reg.
013D	AF	755		XRA A ;Make a 0.
013F	D3 1C	756		OUT ITDB ;Put 0 in the low byte of the MM.
0140	3E 0A	757		MVI A,0AH ;Set up the high byte for divide by 10.
0142	D3 1C	758		OUT ITDB ;Now the RD Timers will work properly.
0144	D3 3A	759	3\$:	OUT CLRBLK ;Unblock the 100 Hz signal.
0146	31 4080	760	INITX:	LXI SP,STACK ;Set up the SP
0149	CD 04DF	761		CALL POWER_UP ;Go do aPower Up.
014C	3E 0B	762		MVI A,SIMMSK ;Mask the Parity and 5.5 Traps
014E	30	763		SIM ;Set the mask
014F	FB	764		EI ;Enable interrupts
0150	AF	765		XRA A ;Make a 0
0151	32 7E10	766		STA RDCON ;Zero the RD Connection flag
0154	32 408A	767		STA COLD ;Zero the Cold Flag
0157	32 40B6	768		STA PUBFLG ;and the Power Up Boot Flag
015A	CD 03C7	769		CALL CTLPRP ;Go do some preping
015D	CD 022E	770		CALL MASK_GENERATOR ;Go generate the mask for USART checking
		771		;
		772		;
		773		;
		774		INIT1
		775		;
		776		The second part of INIT tries to load the Boot Block from the TU-58
		777		;
0160	31 4080	778	INIT1:	LXI SP,STACK ;Prep the SP
0163	21 408D	779		LXI H,SNDBAK ;Point to the Send Packet area
0166	11 06B8	780		LXI D,BOOTPK ;Point to the Boot Packet
0169	06 0E	781		MVI B,14 ;Count for copying
016B	CD 00FB	782		CALL COPY1 ;Go copy it. This copy also sets up the MODEM
		783		;Vector
016E	3A 40BE	784		LDA LR_MASK ;Get the Local/Remote Mask
0171	E6 40	785		ANI 40H ;Check for APT present
0173	3E 00	786		MVI A,0 ;Make a 0 in A
0175	CA 017C	787		JZ INIT2 ;Jump if not APT
0178	3C	788		INR A ;Make a 1
0179	32 40BB	789	INIT0:	STA O_FLAG ;Set the Control 0 flag
017C	21 408C	790	INIT2:	LXI H,RETRY ;Prime the retry count
017F	36 08	791		MVI M,08 ;8 retries allowed
0181	CD 0055	792	INIT3:	CALL INIT_DRIVES ;Go try to init the drive
0184	3A 4116	793		LDA SUCCES ;Check the success code
0187	A7	794		ANA A ;Is it zero?
0188	CA 0190	795		JZ INIT5 ;Jump if it is
018B	2A 40B7	796		LHLD MSGADR ;Get the error message address
018E	F7	797		RST 6 ;Go print it
018F	CF	798		RST 1 ;Go to ROM_IDLE_PREP.
0190	CD 080C	799	INIT5:	CALL RL_VECTOR ;Go load the RAM
0193	3A 4116	800		LDA SUCCES ;Get the success code

Error Addr	Code	Seq	Source statement
0196	A7	801	ANA A ;Check the Success code
0197	C2 01B8	802	JNZ TRYTST ;Jump if no good
019A	3A 4100	803	LDA START ;Get the first instruction in RAM
019D	FE C3	804	CP: 0C3H ;Is it a Jump?
019F	CA 4100	805	JZ START ;Go start up the RAM
01A2	21 06F2	806	LXI H,NOBOOT ;Point to this error message
01A5	F7	807	INIT4: RST 6 ;Go print the message
01A6	21 0700	808	LXI H,DRVMSG ;Point to this message
01A9	F7	809	RST 6 ;Go print it
01AA	3A 4091	810	LDA UNIT ;Get the unit number
01AD	F6 3C	811	ORI 30H ;Make it ASCII
01AF	DF	812	RST 3 ;Go send it
01B0	21 4091	813	LXI H,UNIT ;Point to the Unit number
01B3	35	814	DCR M ;If it was one, this will make it zero
01B4	CA 017C	815	JZ INIT2 ;Go try drive 1
01B7	CF	816	RST 1 ;Go to ROM_IDLE_PREP.
		817	
		818	;.....
		819	
01B8	CD 03E0	820	TRYTST: CALL BRKSUB ;Stop any tape motion
01BB	21 408C	821	LXI H,RETRY ;Get the retry count
01BE	35	822	DCR M ;Decrement it
01BF	C2 0181	823	JNZ INIT3 ;Go try again
01C2	2A 40B7	824	LHLD MSGADR ;Get the error message address
01C5	C3 01A5	825	JMP INIT4 ;Go print the message and check the unit number
		826	
		827	;.....
		828	
		829	
		830	
		831	;.....
		832	
		833	TU_XMIT_READY
		834	
		835	; This routine call TOPREP to zero the Time Out counters. It then loops,
		836	; waiting for TU-58 Xmit Rdy to come up. While waiting it calls GET_CHARACTER
		837	; to allow Control C and P to work and it call TOCHK to time out the TU-58
		838	
		839	TU_XMIT_READY:
01C8	CD 020B	840	CALL TOPREP ;Go prep the Timeouts
01CB	E7	841	1\$: RST 4 ;Go look for Control P or C
01CC	DB 45	842	IN TUSR ;Get the TU-58 Status Register
01CE	1F	843	RAR ;Is Xmit Rdy set?
01CF	D8	844	RC ;Leave if it is
01D0	CD 01F9	845	CALL TOCHK ;Go look for Timeout
01D3	C3 01CB	846	JMP 1\$;Loop back if we come back
		847	
		848	;.....
		849	
		850	

Error Addr	Code	Seq	Source statement
		851	*****
		852	;
		853	;
		854	;
		855	TU_RCVR_DONE:
01D6		856	CALL TOPREP ;Go prep the counters
01D6	CD 020B	857	MVI M,2CH ;Change the outer loop
01D9	36 2C	858	TU_RCVR_0:
01DB		859	RST 4 ;Go look for "Break Out" conditions
01DB	E7	860	IN TUSR ;Get the Status Register
01DC	DB 45	861	ANI 2 ;Look for Rcvr Done
01DE	E6 02	862	JZ 1\$;Jump if not set
01E0	CA 01F3	863	IN TUDB ;Get the character
01E3	DB 44	864	MOV B,A ;Save the character
01E5	47	865	LDA SNDBAK+5 ;See if Handshake has been invoked
01E6	3A 4092	866	ANI 8 ;See if the Handshake bit was set
01E9	E6 08	867	MOV A,B ;Put the character back in A just in case
01EB	78	868	RZ ;Leave if not in Handshake mode
01EC	C8	869	MVI A,10H ;Get a Continue
01ED	3E 10	870	OUT TUDB ;Send it out
01EF	D3 44	871	MOV A,B ;Get the character
01F1	78	872	RET ;Leave
01F2	C9	873	1\$: CALL TOCHK ;Go check for a Timeout
01F3	CD 01F9	874	JMP TU_RCVR_0 ;Loop back
01F6	C3 01DB	875	;
		876	*****
		877	;
		878	;
		879	*****
		880	;
		881	;
		882	;
		883	;
		884	;
		885	;
		886	;
		887	;
		888	;
		889	;
		890	;
		891	;
		892	;
		893	TOCHK: LXI H,TOCNT ;Point to the inner loop count
01F9	21 4083	894	DCR M ;Decrement it
01FC	35	895	RNZ ;Leave if not zero
01FD	C0	896	LHLD TOCNT1 ;Get the word wide count
01FE	2A 4084	897	MOV A,H ;Put the high byte in the accumulator
0201	7C	898	ORA L ;OR in the low byte
0202	B5	899	CZ TUERR2 ;Go report the error if zero
0203	CC 0079	900	DCX H ;Reduce the count
0206	2B		

Error	Addr	Code	Seq	Source statement
	0207	22 4084	901	SHLD TOCNT1 ;Save the new value
	020A	C9	902	RET ;Leave
			903	;
			904	;.....
			905	;
			906	;
			907	;.....
			908	;
			909	TOPREP
			910	;
			911	This routine zeros the 8 bit and 16 bit counters, TOCNT and TOCNT1.
			912	;
020B	21 4083		913	TOPREP: LXI H,TOCNT ;Point to the inner loop count
020E	AF		914	XRA A ;Make a zero in the Accumulator
020F	77		915	MOV M,A ;Send it out
0210	23		916	INX H ;Bump the pointer
0211	77		917	MOV M,A ;Zero the high byte of the inner loop counter
0212	23		918	INX H ;Point to the outer loop counter
0213	77		919	MOV M,A ;Zero the it
0214	C9		920	RET ;Leave
			921	;
			922	;.....
			923	;
			924	;
			925	;
			926	;
			927	;.....
			928	;
0215			929	MODEM_CHECK:
0215	CD 068D		930	CALL MODEM_ENTRY ;Go attempt to sequence up the modem.
0218	21 7E10		931	LXI H,RDCON ;Point to the RD connect flag
021B	7E		932	MOV A,M ;Get it
021C	23		933	INX H ;Bump the pc nter to the tracking copy
021D	BE		934	CMP M ;Are they the same?
021E	CA 0225		935	JZ 1\$;Jump if the same
0221	A7		936	ANA A ;Is it now a 1?
0222	C2 022E		937	JNZ MASK_GENERATOR ;Go fix the mask if it is
0225	21 4081		938	1\$: LXI H,SWITCH_POS ;Point to the Switch Position flag
0228	DB 06		939	IN REMOTE ;Get the Remote/Local Switch
022A	E6 01		940	ANI 01 ;Save the LSB only
022C	BE		941	CMP M ;Has anything changed?
022D	C8		942	RZ ;IF NO CHANGE, leave.
			943	;
			944	MASK_GENERATOR:
022E			945	LXI H,LR_MASK ;Point to the Console Mode Mask
0231	DB 06		946	IN REMOTE ;Get the Remote Bit
0233	1F		947	RAR ;Check for set
0234	D2 024B		948	JNC 2\$;Jump if in Remote
0237	3E 1		949	MVI A,1 ;Get a one
0239	32 4081		950	STA SWITCH_POS ;Set this flag to Local

Error	Addr	Code	Seq	Source statement
023C	36 81	951	1\$:	MVI M,81H ;Set up Console Mode mask
023E	DB 04	952		IN APTFLG ;Check for APT
0240	1F	953		RAR ;Is it there?
0241	D8	954		RC ;Leave if not
0242	3E 15	955		MVI A,15H ;Get this constant
0244	D3 43	956		OUT TRCR ;Negate the Modem Control bits
0246	AF	957		XRA A ;Make a 0
0247	32 7E04	958		STA TALK_FLAG ;Zero the Talk Flag
024A	C9	959		RET ;Leave
024B	AF	960	2\$:	XRA A ;Make a zero
024C	32 4081	961		STA SWITCH_POS ;Zero this flag to indicate Remote position
024F	DB 04	962		IN APTFLG ;Get the APT present flag
0251	1F	963		RAR ;Check it
0252	DA 025C	964		JC 3\$;Jump if not APT
0255	36 42	965		MVI M,42H ;Set it up for APT only
0257	3E 37	966		MVI A,37H ;Assert the Modem Control bits
0259	D3 43	967		OUT TRCR ;Do it
025B	C9	968		RET ;Leave
025C	3A 7E10	969	3\$:	LDA RDCON ;Get this copy of the RD Connect Flag
025F	32 7E11	970		STA RDCON1 ;Save the updated version
0262	A7	971		ANA A ;Check it
0263	CA 023C	972		JZ 1\$;Set up Local if not set
0266	36 24	973		MVI M,24H ;Set up Fully Disabled mode.
0268	AF	974		XRA A ;Make a zero
0269	32 7E00	975		STA REMRSR ;Clear this done bit
026C	3C	976		INR A ;Make a 1
026D	32 7E02	977		STA REMXSR ;Set this Ready bit
0270	7E13	978		STA TALKSR ;and this one
0273	C9	979		RET ;Leave
		980	:	
		981	:	
		982	:	
		983	:	
		984	:	
		985	:	
		986	:	
		987	:	
		988	:	
		989	:	
		990	:	ROM_IDLE_PREP
		991	:	
0274		992	:	ROM_IDLE_PREP:
0274	31 4080	993		LXI SP,STACK ;Fix the Stack Pointer
0277	AF	994		XRA A ;Make a 0
0278	32 40FE	995		STA CTL_CHR_DISABLE ;Turn off the Control Character Disable flag
027B	32 408B	996		STA O_FLAG ;and the Control O flag
027E	21 0708	997		LXI H,RPRMPT ;Point to the Prompt
0281	F7	998		RST 6 ;Go print it
0282	21 7000	999		LXI H,CHRCNT ;Point to the Character Count location
0285	36 50	1000		MVI M,80 ;Set the maximum count

Error	Addr	Code	Seq	Source statement
	0287	23	1001	INX H ;Point to the beginning of the buffer
	0288	22 40CA	1002	SHLD BUFADR ;Save the address
	028B	AF	1003	XRA A ;Make a 0
	028C	32 4082	1004	STA BYTSUM ;Zero out this area
			1005	;
			1006	;.....
			1007	
			1008	
			1009	;.....
			1010	;
	028F		1011	ROM_IDLE:
	028F	CD 03F1	1012	CALL GET_SILO ;Go look for characters
	0292	D2 028F	1013	JNC ROM_IDLE ;Loop if no character
	0295	3A 7E04	1014	LDA TALK_FLAG ;Get the Talk Flag
	0298	1F	1015	RAR ;Is it set?
	0299	D2 02A2	1016	JNC 1\$;Jump if not
	029C	CD 080F	1017	CALL TALK_VECTOR ;Go to the Talk routine
	029F	C3 028F	1018	JMP ROM_IDLE ;Loop back when done
	02A2	78	1019	1\$: MOV A,B ;Get the character for a bit
	02A3	E6 7F	1020	ANI 7FH ;Clear the MSB
	02A5	47	1021	MOV B,A ;Put it back in B
	02A6	21 7000	1022	LXI H,CHRCNT ;Point to the count
	02A9	7E	1023	MOV A,M ;Get it
	02AA	A7	1024	ANA A ;Is it zero?
	02AB	78	1025	MOV A,B ;Get the character into A
	02AC	C2 02B4	1026	JNZ 2\$;Jump if count not 0
	02AF	FE 0D	1027	CPI CR ;Look for Carriage Return
	02B1	C2 028F	1028	JNZ ROM_IDLE ;Loop back if not
	02B4	F5	1029	2\$: PUSH PSW ;Save a copy of the character for later.
	02B5	35	1030	DCR M ;Reduce the count
	02B6	2A 40CA	1031	LHLD BUFADR ;Get the pointer
	02B9	70	1032	MOV M,B ;Store the character
	02BA	23	1033	INX H ;Bump the pointer
	02BB	22 40CA	1034	SHLD BUFADR ;Save the new address
	02BE	3A 40BE	1035	LDA LR_MASK ;Get the Local/Remote mask
	02C1	E6 40	1036	ANI 40H ;See if APT there
	02C3	C2 02C8	1037	JNZ 3\$;Jump if it is
	02C6	78	1038	MOV A,B ;Put the character in A
	02C7	DF	1039	RST 3 ;Go send it back
	02C8	F1	1040	3\$: POP PSW ;Get the character back
	02C9	FE 0D	1041	CPI CR ;Is it a carriage return?
	02CB	C2 028F	1042	JNZ ROM_IDLE ;Loop back if not
			1043	
	02CE	21 7001	1044	LXI H,CHRBUF ;Fix up the pointer
	02D1	7E	1045	MOV A,M ;Get the first character
	02D2	FE 58	1046	CPI 'X' ;Is it an Xfer command?
	02D4	C2 0338	1047	JNZ NOT_X_COMMAND ;If it isn't, leave
	02D7	3E 01	1048	MVI A,1 ;Make a one
	02D9	32 40FE	1049	STA CTL_CHR_DISABLE ;Disable Control Character checking
	02DC	CD 03F1	1050	4\$: CALL GET_SILO ;Go get the checksum

B	1	COVER	PAGE
C	1	COVER	PAGE
D	1	ENOAD	LIS
E	1	ENOAD	LIS
F	1	ENOAD	LIS
G	1	ENOAD	LIS
H	1	ENOAD	LIS
I	1	ENOAD	LIS
J	1	ENOAD	LIS
K	1	ENOAD	LIS
L	1	ENOAD	LIS
M	1	ENOAD	LIS
N	1	ENOAD	LIS
B	2	ENOAD	LIS
C	2	ENOAD	LIS
D	2	ENOAD	LIS
E	2	ENOAD	LIS
F	2	ENOAD	LIS
G	2	ENOAD	LIS
H	2	ENOAD	LIS
I	2	ENOAD	LIS
J	2	ENOAD	LIS
K	2	ENOAD	LIS
L	2	ENOAD	LIS
M	2	ENOAD	LIS
N	2	ENOAD	LIS
B	3	ENOAD	LIS
C	3	ENOAD	LIS
D	3	ENOAD	LIS
E	3	ENOAD	LIS
F	3	ENOAD	LIS
G	3	ENOAD	LIS
H	3	ENOAD	LIS
I	3	ENOAD	LIS
J	3	ENOAD	LIS
K	3	ENOAD	LIS
L	3	ENOAD	LIS
M	3	ENOAD	LIS
N	3	ENOAD	LIS
B	4	ENOAD	LIS
C	4	ENOAD	LIS
D	4	ENOAD	LIS
E	4	ENOAD	LIS
F	4	ENOAD	LIS
G	4	ENOAD	LIS
H	4	ENOAD	LIS
I	4	ENOAD	LIS
J	4	ENOAD	LIS
K	4	ENOAD	LIS
L	4	ENOAD	LIS
M	4	ENOAD	LIS
N	4	ENOAD	LIS
B	5	ENOAD	LIS
C	5	ENOAD	LIS
D	5	ENOAD	LIS
E	5	ENOAD	LIS
F	5	ENOAD	LIS
G	5	ENOAD	LIS
H	5	ENOAD	LIS
I	5	ENOAD	LIS
J	5	ENOAD	LIS
K	5	ENOAD	LIS
L	5	ENOAD	LIS
M	5	ENOAD	LIS
N	5	ENOAD	LIS
B	6	ENOAD	LIS
C	6	ENOAD	LIS
D	6	ENOAD	LIS
E	6	ENOAD	LIS

F	6	ENOAD	LIS
G	6	ENOAD	LIS
H	6	ENOAD	LIS
I	6	ENOAD	LIS
J	6	ENOAD	LIS
K	6	ENOAD	LIS
L	6	ENOAD	LIS
M	6	ENOAD	LIS
N	6	ENOAD	LIS
B	7	ENOAD	LIS
C	7	ENOAD	LIS
D	7	ENOAD	LIS
E	7	ENOAD	LIS
F	7	ENOAD	LIS
G	7	ENOAD	LIS
H	7	ENOAD	LIS
I	7	ENOAD	LIS
J	7	ENOAD	LIS
K	7	ENOAD	LIS
L	7	ENOAD	LIS
M	7	ENOAD	LIS
N	7	ENOAD	LIS
B	8	ENOAD	LIS
C	8	ENOAD	LIS
D	8	ENOAD	LIS
E	8	ENOAD	LIS
F	8	ENOAD	LIS
G	8	ENOAD	LIS
H	8	ENOAD	LIS
I	8	ENOAD	LIS
J	8	ENOAD	LIS
K	8	ENOAD	LIS
L	8	ENOAD	LIS
M	8	ENOAD	LIS
N	8	ENOAD	LIS
B	9	ENOAD	LIS
C	9	ENOAD	LIS
D	9	ENOAD	LIS
E	9	ENOAD	LIS
F	9	ENOAD	LIS
G	9	ENOAD	LIS
H	9	ENOAD	LIS
I	9	ENOAD	LIS
J	9	ENOAD	LIS
K	9	ENOAD	LIS
L	9	ENOAD	LIS
M	9	ENOAD	LIS
N	9	ENOAD	LIS
B	10	ENOAD	LIS
C	10	ENOAD	LIS
D	10	ENOAD	LIS
E	10	ENOAD	LIS
F	10	ENOAD	LIS
G	10	ENOAD	LIS
H	10	ENOAD	LIS
I	10	ENOAD	LIS
J	10	ENOAD	LIS
K	10	ENOAD	LIS
L	10	ENOAD	LIS
M	10	ENOAD	LIS
N	10	ENOAD	LIS
B	11	ENOAD	LIS
C	11	ENOAD	LIS
D	11	ENOAD	LIS
E	11	ENOAD	LIS
F	11	ENOAD	LIS
G	11	ENOAD	LIS
H	11	ENOAD	LIS
I	11	ENOAD	LIS

J	11	ENOAD	LIS
K	11	ENOAD	LIS
L	11	ENOAD	LIS
M	11	ENOAD	LIS
N	11	ENOAD	LIS
B	12	ENOAD	LIS
C	12	ENOAD	LIS
D	12	ENOAD	LIS
E	12	ENOAD	LIS
F	12	ENOAD	LIS
G	12	ENOAD	LIS
H	12	ENOAD	LIS
I	12	ENOAD	LIS
J	12	ENOAD	LIS
K	12	ENOAD	LIS
L	12	ENOAD	LIS
M	12	ENOAD	LIS
N	12	ENOAD	LIS
B	13	ENOAD	LIS
C	13	ENOAD	LIS
D	13	ENOAD	LIS
E	13	ENOAD	LIS
F	13	ENOAD	LIS
G	13	ENOAD	LIS
H	13	ENOAD	LIS
I	13	ENOAD	LIS
J	13	ENOAD	LIS
K	13	ENOAD	LIS
L	13	ENOAD	LIS
M	13	ENOAD	LIS
N	13	ENOAD	LIS
B	14	ENOAD	LIS
C	14	ENOAD	LIS
D	14	ENOAD	LIS
E	14	ENOAD	LIS
F	14	ENOAD	LIS
G	14	ENOAD	LIS
H	14	ENOAD	LIS
I	14	ENOAD	LIS
J	14	ENOAD	LIS
K	14	ENOAD	LIS
L	14	ENOAD	LIS
M	14	ENOAD	LIS
N	14	ENOAD	LIS
B	15	ENOAD	LIS
C	15	ENOAD	LIS
D	15	ENOAD	LIS
E	15	ENOAD	LIS
F	15	ENOAD	LIS
G	15	ENOAD	LIS
H	15	ENOAD	LIS
I	15	ENOAD	LIS
J	15	ENOAD	LIS
K	15	ENOAD	LIS
L	15	ENOAD	LIS
M	15	ENOAD	LIS
N	15	ENOAD	LIS
B	16	ENOAD	LIS
C	16	ENOAD	LIS
D	16	ENOAD	LIS
E	16	ENOAD	LIS
F	16	ENOAD	LIS
G	16	ENOAD	LIS
H	16	ENOAD	LIS
I	16	ENOAD	LIS
J	16	ENOAD	LIS
K	16	ENOAD	LIS
L	16	ENOAD	LIS

Error	Addr	Code	Seq	Source statement
	02DF	D2 02DC	1051	JNC 4\$;Loop until a character comes in
	02E2	3A 4082	1052	LDA BYTSUM ;Let's check it
	02E5	A7	1053	ANA A ;Is it OK?
	02E6	C2 033D	1054	JNZ ROM_ERR ;Jump if Command Checksum bad.
	02E9		1055	X_CMND_0:
	02E9	21 0708	1056	LXI H,RPRMPT ;Point to the ROM Prompt
	02EC	F7	1057	RST 6 ;Go send it
	02ED	AF	1058	XRA A ;Make a 0
	02EE	32 4082	1059	STA BYTSUM ;Zero the Byte Checksum
	02F1	21 4100	1060	LXI H,4100H ;Point to the part of RAM where we start loading
	02F4	22 40C4	1061	SHLD XADRES ;Save the address
	02F7	CD 03F1	1062	1\$: CALL GET_SILO ;Go look for the next character
	02FA	D2 02F7	1063	JNC 1\$;Loop until one comes in
	02FD	2A 40C4	1064	LHLD XADRES ;Get the pointer
	0300	7C	1065	MOV A,H ;Get the high byte of the address
	0301	FE 7E	1066	CPI 7EH ;Look for beginning of RD area
	0303	D2 0311	1067	JNC 3\$;Jump if it is
	0306	FE 4B	1068	CPI 4BH ;See if it is 4BH
	0308	C2 0310	1069	JNZ 2\$;Jump if not
	030B	7D	1070	MOV A,L ;Get the low byte
	030C	17	1071	RAL ;See if it is 80H or greater
	030D	DA 0311	1072	JC 3\$;Jump if it is
	0310	70	1073	2\$: MOV M,B ;Send it out
	0311	23	1074	3\$: INX H ;Bump the pointer
	0312	22 40C4	1075	SHLD XADRES ;Store the new value
	0315	7C	1076	MOV A,H ;Get the high byte
	0316	17	1077	RAL ;Look for 80 in the high byte
	0317	D2 02F7	1078	JNC 1\$;Loop back if not there
	031A	CD 03F1	1079	4\$: CALL GET_SILO ;Go get the checksum
	031D	D2 031A	1080	JNC 4\$;Wait for the character
	0320	3A 4082	1081	LDA BYTSUM ;Let's check it
	0323	A7	1082	ANA A ;Is it good
	0324	21 070E	1083	LXI H,NACK_X_DATA ;Point to the Error message just in case.
	0327	C2 0340	1084	JNZ ROM_ERR1 ;Jump if Data Checksum bad.
	032A	3A 4103	1085	LDA GOSV ;Check the OP code in this byte
	032D	FE C3	1086	CPI 0C3H ;This it should be a JMP
	032F	CA 4103	1087	JZ GOSV ;Use this to get to the proper place in the RAM
	0332	21 0713	1088	LXI H,VALIDITY_ERR ;Point to this error message.
	0335	C3 0340	1089	JMP ROM_ERR1 ;Go print it.
			1090	;
			1091	;*****
			1092	
			1093	
			1094	
			1095	;*****
			1096	;
0338			1097	NOT_X_COMMAND:
0338	FE 0D		1098	CPI CR ;Look for Carriage return
033A	CA 0274		1099	JZ ROM_IDLE_PREP ;Leave with printing an error message.
033D			1100	ROM_ERR:

Error Addr	Code	Seq	Source statement
033D	21 0704	1101	LXI H,SYNTAX ;Point to the error message
0340		1102	ROM_ERR1:
0340	AF	1103	XRA A ;Make a 0
0341	32 40FE	1104	STA CTL_CHR_DISABLE ;Turn off the disable (which may or may not
		1105	;be set).
0344	F7	1106	RST 6 ;Go to LPRINT
0345	CF	1107	RST 1 ;Go back to Limited Console
		1108	;
		1109	;*****
		1110	
		1111	
		1112	
		1113	;*****
		1114	;
		1115	; This routine:
		1116	;
		1117	; Tests for control characters.
		1118	; Enters valid characters into the Silo (if possible).
		1119	;
		1120	;
0346		1121	GET_LOCAL:
0346	3A 7E04	1122	LDA TALK_FLAG ;Get the Talk Flag
0349	E6 01	1123	ANI 1 ;Save only the low bit.
034B		1124	GET_LOCA0:
034B	21 40D3	1125	LXI H,RUN_FLAG ;Point to the Run Flag
034E	B6	1126	ORA M ;OR them together
034F	C2 038B	1127	JNZ 2\$;IF (PROGRAM MODE + TALK MODE), jump.
0352	79	1128	MOV A,C ;ELSE, get the correct SR back
0353	E6 20	1129	ANI 20H ;Look for Framing Error set
0355	CA 0363	1130	JZ 1\$;IF NOT FRAMING ERROR, jump.
0358	78	1131	MOV A,B ;Get the character
0359	A7	1132	ANA A ;Is it zero?
035A	C2 0363	1133	JNZ 1\$;IF NOT ZERO, jump.
035D	21 06CC	1134	LXI H,BRKMSG ;ELSE, point to the Break Message
0360	C3 05AF	1135	JMP CNTLPO ;Go share the finish up code
0363	3A 40FE	1136	1\$: LDA CTL_CHR_DISABLE ;Get the Control Character Disable Flag
0366	1F	1137	RAR ;Put it in the carry
0367	DA 038B	1138	JC 2\$;IF CONTROL CHARACTER DISABLED, jump.
036A	78	1139	MOV A,B ;ELSE, get the character
036B	E6 7F	1140	ANI 7FH ;Mask off the parity bit
036D	FE 14	1141	CPI 14H ;Do a quick range check. All the Control
		1142	;Characters we care about are less than 14 Hex.
036F	D2 038B	1143	JNC 2\$;IF Char => 14H, jump.
0372	FE 13	1144	CPI CNTRL5 ;ELSE, look for Control S
0374	CA 03B9	1145	JZ SET_S_FLAG ;IF CONTROL S, jump.
0377	FE 11	1146	CPI CNTRLQ ;ELSE, look for a Control Q
0379	CA 03C3	1147	JZ CLEAR_S_FLAG ;IF CONTROL Q, jump.
037C	FE 0F	1148	CPI CNTRLO ;ELSE, look for a control O
037E	CA 051D	1149	JZ FLIP_O_FLAG ;IF CONTROL Q, jump.
0381	FE 03	1150	CPI CNTRLC ;ELSE, check for Control C

Error	Addr	Code	Seq	Source statement
0383	CA 054C	1151	JZ	CNTLC ;IF CONTROL C, jump.
0386	FE 10	1152	CPI	CNTRLP ;ELSE, check for Control P.
0388	CA 05AC	1153	JZ	CNTLP ;IF CONTROL P, jump.
038B	3A 40D0	1154	LDA	SILO_ACTIVE ;Get the silo flag
038E	1F	1155	RAR	;Put it in the carry
038F	3A 40D1	1156	LDA	SILO_IN_INDX ;Get the Silo Input Index
0392	D2 039A	1157	JNC	3\$;Jump if silo empty
0395	21 40D2	1158	LXI	H,SILO_OUT_INDX ;Point to the Silo Output Index
0398	BE	1159	CMP	M ;Are they the same?
0399	C8	1160	RZ	;Leave with carry clear if they are
039A	5F	1161	MOV	E,A ;Put the Index in E
039B	21 4B80	1162	LXI	H,CHAR_SILO ;Point to the Character Silo
039E	85	1163	ADD	L ;Add the Address Index
039F	6F	1164	MOV	L,A ;Put the adjusted address in L
03A0	70	1165	MOV	M,B ;Store the character
03A1	F6 40	1166	ORI	40H ;Alter the address to point into the Flag silo
03A3	6F	1167	MOV	L,A ;Put the altered low byte in L
03A4	71	1168	MOV	M,C ;Store the Source Flag and the SR
03A5	7B	1169	MOV	A,E ;Get the Index
03A6	3C	1170	INR	A ;Bump it
03A7	E6 3F	1171	ANI	3FH ;Force the wrap around
03A9	32 40D1	1172	STA	SILO_IN_INDX ;Save the updated value
03AC	3E 01	1173	MVI	A,1 ;Make a 1
03AE	32 40D0	1174	STA	SILO_ACTIVE ;Set the Silo Active Flag
03B1		1175	FINISH_LOCAL:	
03B1	79	1176	MOV	A,C ;Get the source flag
03B2	17	1177	RAL	;Is this an RD character?
03B3	D0	1178	RNC	;Leave if not an RD character
03B4	AF	1179	XRA	A ;Make a 0
03B5	32 7E00	1180	STA	REMRSR ;Zero the Remote Done bit
03B8	C9	1181	RET	;Leave
		1182		;
		1183		*****
		1184		
		1185		
		1186		
		1187		*****
		1188		;
03B9		1189	SET_S_FLAG:	
03B9	3E 01	1190	MVI	A,1 ;Make a 1
03BB		1191	SET_S_FLO:	
03BB	32 40CD	1192	STA	S_FLAG ;Set the S Flag
03BE	CD 03B1	1193	CALL	FINISH_LOCAL ;Go clear the RD Rcv Done if set
03C1	AF	1194	XRA	A ;Clear the carry
03C2	C9	1195	RET	;Leave
		1196		
03C3		1197	CLEAR_S_FLAG:	
03C3	AF	1198	XRA	A ;Make a 0
03C4	C3 03BB	1199	JMP	SET_S_FLO ;Go share code
		1200		;

Error	Addr	Code	Seq	Source statement
			1201	;*****
			1202	
			1203	
			1204	;*****
			1205	
03C7	AF		1206	CTLPRP: XRA A ;Make a 0
03C8	32 7E46		1207	STA MSG15_COUNT ;Zap this count
03CB	32 409D		1208	STA INDFLG ;Turn off this flag
03CE	32 40FE		1209	STA CTL_CHR_DISABLE ;and this one also
03D1	32 40FF		1210	STA NOCHK ;and this one as well
03D4	32 40D4		1211	STA NOTYPE ;and another
03D7	32 40D0		1212	STA SILO_ACTIVE ;one more
03DA	32 40D2		1213	STA SILO_OUT_INDX ;Zero the Silo Output Index
03DD	32 40D1		1214	STA SILO_IN_INDX ;and the Silo Input Index
03E0	3E 0D		1215	BRKSUB: MVI A,BREAK ;Put the value of a Break into A
03E2	D3 47		1216	OUT TUCR ;and send it to the Command register
03E4	06 00		1217	MVI B,0 ;Put a 0 in B
03E6	05		1218	1\$: DCR B ;Decrement the count
03E7	C2 03E6		1219	JNZ 1\$;Loop if not done
03EA	3E 15		1220	MVI A,15H ;Put the normal Command Register value in A
03EC	D3 47		1221	OUT TUCR ;Send it to the Command Register
03EE	DB 44		1222	IN TUDB ;Read the DB to clear out the garbage
03F0	C9		1223	RET ;Leave
			1224	
			1225	;*****
			1226	
			1227	
			1228	;*****
			1229	
03F1			1230	GET_SILO:
03F1	CD 040F		1231	CALL READ_SILO ;Go read the silo
03F4	D0		1232	RNC ;Leave if no character found
03F5	7B		1233	MOV A,E ;Get the Silo Output Index.
03F6	3C		1234	INR A ;Bump it
03F7	E6 3F		1235	ANI 3FH ;Save only the Modulo 40H count
03F9	32 40D2		1236	STA SILO_OUT_INDX ;Store it
03FC	21 40D1		1237	LXI H,SILO_IN_INDX ;Point to the Silo Output Index.
03FF	BE		1238	CMP M ;See if the addresses are now equal
0400	C2 0407		1239	JNZ 1\$;IF NOT EQUAL, jump.
0403	AF		1240	XRA A ;ELSE, make a 0
0404	32 40D0		1241	STA SILO_ACTIVE ;and clear the Silo Active flag
0407	78		1242	1\$: MOV A,B ;Put the character in A
0408	21 4082		1243	LXI H,BYTSUM ;Point to the Byte Checksum area
040B	86		1244	ADD M ;Add it in
040C	77		1245	MOV M,A ;Store the result
040D	37		1246	STC ;Set the carry
040E	C9		1247	RET ;Leave
			1248	
040F			1249	READ_SILO:
040F	CD 0427		1250	CALL GET_CHARACTER ;Go look for characters

Error	Addr	Code	Seq	Source statement	
0412	3A 40D0	1251	LDA	SILO_ACTIVE	;Get the Silo Active Flag
0415	1F	1252	RAR		;Is it set?
0416	D0	1253	RNC		;Leave with carry clear if not
0417	3A 40D2	1254	LDA	SILO_OUT_IND	;Get the Silo Output Index
041A	5F	1255	MOV	E,A	;Save a copy in E
041B	21 4B80	1256	LXI	H,CHAR_SILO	;Point to the Character Silo
041E	85	1257	ADD	L	;Add the Index
041F	6F	1258	MOV	L,A	;Put the low byte of the address into L
0420	46	1259	MOV	B,M	;Get the character
0421	F6 40	1260	ORI	40H	;Alter the low byte of the address
0423	6F	1261	MOV	L,A	;Put it in L
0424	4E	1262	MOV	C,M	;Get the source flag and the SR
0425	37	1263	STC		;Set the carry to indicate character present
0426	C9	1264	RET		;Leave
		1265			
		1266			
		1267			
		1268			
		1269			
		1270			
0427		1271		GET_CHARACTER:	
0427	3A 40BE	1272	LDA	LR_MASK	;Get the Local/Remote Mask
042A	E6 40	1273	ANI	40H	;Look for APT in control
042C	C2 049F	1274	JNZ	GET_APT	;Jump if it is
042F	CD 0215	1275	CALL	MODEM_CHECK	;Go check out the Modem stuff
0432	DB 03	1276	IN	SECURE	;Look at the Secure bit of the Keyswitch.
0434	21 40D3	1277	LXI	H,RUN_FLAG	;Point to the Run Flag
0437	B6	1278	ORA	M	;OR them together
0438	21 7E04	1279	LXI	H,TALK_FLAG	;Point to the Talk Mode Flag
043B	B6	1280	ORA	M	;OR it in
043C	1F	1281	RAR		;Put the result in the carry
043D	D0	1282	RNC		;Leave with carry clear if DISABL*(-RUN)*(-TALK)
043E	3A 40BE	1283	LDA	LR_MASK	;Get the Local/Remote Mask
0441	FE A5	1284	CPI	0A5H	;Look for Parallel Control
0443	C2 044E	1285	JNZ	1\$	;Jump if not
0446	3A 7E10	1286	LDA	RDCON	;Get the RD Connected flag
0449	21 7E05	1287	LXI	H,DISCARD_RO	;Point to the Discard Remote Output bit
044C	B6	1288	ORA	M	;OR them together
044D	C8	1289	RZ		;Leave if (-RDCON*-DISCARD_RO)
044E	DB 49	1290	IN	TTSR	;Get the Local SR
0450	E6 02	1291	ANI	2	;Look for Receiver Done
0452	21 7E00	1292	LXI	H,REMRSR	;Point to the Remote Pseudo SR
0455	B6	1293	ORA	M	;OR it in
0456	C8	1294	RZ		;Leave with carry clear if nothing set
0457	DB 49	1295	IN	TTSR	;Get the Local SR
0459	4F	1296	MOV	C,A	;Save a copy in C
045A	E6 02	1297	ANI	RCVDON	;Look for Receiver Done
045C	CA 048A	1298	JZ	3\$	;Jump if not
045F	DB 48	1299	IN	TTDB	;Get the character
0461	47	1300	MOV	B,A	;Put it in B

Error	Addr	Code	Seq	Source statement	
0462	79		1301	MOV A,C	;Get the SR back
0463	E6 3F		1302	ANI 3FH	;Clear bits 7 and 6 to indicate Local character
0465	4F		1303	MOV C,A	;Put it back in C
0466	E6 38		1304	ANI 38H	;Look for errors
0468	CA 046F		1305	JZ 2\$	;Jump if none
046B	3E 35		1306	MVI A,TTCR_CODE	;RTS (DSRS for Europe RD), Error Reset, Xmit
			1307		;and Rcv Enable
046D	D3 4B		1308	OUT TTCR	;Reset the errors
046F	3A 7E04		1309	LDA TALK_FLAG	;Are we in Talk Mode?
0472	1F		1310	RAR	;Check the LSB.
0473	DA 0346		1311	JC GET_LOCAL	;IF TALK MODE, THEN jump.
0476	3A 40BE		1312	LDA LR_MASK	;ELSE, get the Local/Remote Mask
0479	17		1313	RAL	;Check for Local enabled
047A	DA 0346		1314	JC GET_LOCAL	;IF LOCAL CONTROL ENABLED, THEN take care of
			1315		;things the normal way.
047D	78		1316	MOV A,B	;ELSE, get the character back into A.
047E	E6 7F		1317	ANI 7FH	;Zap the parity bit.
0480	FE 03		1318	CPI CNTRLC	;Check for a control C.
0482	C2 048A		1319	JNZ 3\$	;IF NOT CONTROL C, THEN jump.
0485	3E 40		1320	MVI A,40H	;ELSE, generate the INTR constant.
0487	32 7E70		1321	STA CTL_C_INTR	;Store it here.
048A	3A 40BE		1322	LDA LR_MASK	;Get the LR_MASK again
048D	E6 20		1323	ANI 20H	;Look for Remote enabled
048F	C8		1324	RZ	;If not, leave with carry clear
0490	21 7E00		1325	LXI H,REMRSR	;Point to the Remote Pseudo SR
0493	7E		1326	MOV A,M	;Get it
0494	E6 02		1327	ANI RCVDON	;Check for Receiver Done
0496	C8		1328	RZ	;Leave with carry clear if not Done
0497	F6 80		1329	ORI 80H	;Set the Remote character flag
0499	4F		1330	MOV C,A	;Put the copy of the SR in C
049A	23		1331	INX H	;Point to the character
049B	46		1332	MOV B,M	;and get it
049C	C3 0346		1333	JMP GET_LOCAL	;Go share the character processing code
			1334		
049F			1335	GET_APT:	
049F	21 4081		1336	LXI H,SWITCH_POS	;Point to the Switch Position flag
04A2	DB 06		1337	IN REMOTE	;Get the Remote/Local Switch
04A4	E6 01		1338	ANI 01	;Save the LSB only
04A6	BE		1339	CMP M	;Has anything changed?
04A7	C4 022E		1340	CNZ MASK_GENERATOR	;Go fix up the Mask if anything has
04AA	DB 03		1341	IN SECURE	;Look at the Secure bit of the Keyswitch.
04AC	21 40D3		1342	LXI H,RUN_FLAG	;Point to the Run Flag
04AF	B6		1343	ORA M	;OR them together
04B0	1F		1344	RAR	;Put the result in the carry
04B1	D0		1345	RNC	;Leave if DISABL*(-RUN)
04B2	DB 41		1346	IN TRSR	;Get the Remote SR
04B4	4F		1347	MOV C,A	;Put a copy of the SR in C
04B5	E6 02		1348	ANI RCVDON	;Check for Receiver Done
04B7	C8		1349	RZ	;Leave with carry clear if Done not set
04B8	DB 40		1350	IN TRDB	;Get the character

Error	Addr	Code	Seq	Source statement
	04BA	47	1351	MOV B,A ;Put it in B
	04BB	79	1352	MOV A,C ;Get the SR
	04BC	E6 3F	1353	ANI 3FH ;Clear bits 7 and 6
	04BE	F6 40	1354	ORI 40H ;Set bit 6 to indicate APT character
	04C0	4F	1355	MOV C,A ;Put it back in C
	04C1	E6 38	1356	ANI 38H ;Check for errors
	04C3	CA 034B	1357	JZ GET_LOCA0 ;Skip next part if no errors
	04C6	3E 37	1358	MVI A,37H ;Prepare to reset errors
	04C8	D3 43	1359	OUT TRCR ;Do it
	04CA	AF	1360	XRA A ;Clear the Accumulator
	04CB	C3 034B	1361	JMP GET_LOCA0 ;Go share cleanup code
			1362	;
			1363	*****
			1364	;
			1365	*****
			1366	;
			1367	;
	04CE	D3 32	1368	ACLCHK: OUT SETACL ;Assert ACLO
	04D0	F5	1369	PUSH PSW ;Save the AC and the PSW
	04D1	3E 0D	1370	MVI A,BREAK ;Sending this will stop the Tape if it's moving
	04D3	D3 47	1371	OUT TUCR ;Send it
	04D5	3A 4080	1372	LDA RAM_VALID ;See if there is RAM code to jump to
	04D8	A7	1373	ANA A ;Is this byte 0?
	04D9	C2 4120	1374	JNZ ACLO ;Jump to the RAM code if not
	04DC	C3 0146	1375	JMP INITX ;Go start the power up sequence
			1376	;
			1377	*****
			1378	;
			1379	*****
			1380	;
			1381	;
	04DF		1382	POWER_UP:
	04DF	D3 32	1383	OUT SETACL ;Assert ACLO
	04E1	01 014E	1384	LXI B,334 ;Set up a 2 ms count
	04E4	0B	1385	1\$: DCX B ;Count it down
	04E5	79	1386	MOV A,C ;Get the low byte into A
	04E6	B0	1387	ORA B ;OR in the high
	04E7	C2 04E4	1388	JNZ 1\$;Loop until done
	04EA		1389	POWER_0:
	04EA	D3 2F	1390	OUT SETBSY ;Set Bus Busy
	04EC	F5	1391	PUSH PSW ;Do this Push and Pop to guarantee a 5 usec min
	04ED	F1	1392	POP PSW ;between asserting Bus Busy and DCLO
	04EE	D3 30	1393	1\$: OUT SETDCL ;Assert DCLO
	04F0	D3 35	1394	OUT INITH ;Assert INIT
	04F2	D3 31	1395	OUT CLRDCL ;Negate DCLO
	04F4	01 1F13	1396	LXI B,7955 ;Set up for a 70 Msec delay
	04F7	CD 0512	1397	CALL WAIT ;Need this to allow 5 usec after releasing DC LO
	04FA	DA 04EE	1398	JC 1\$;Loop back if DCL set
	04FD	D3 34	1399	OUT INITL ;Negate INIT

**	04FF	C3 1807		JMP V12FIX ;V12 Jump to subroutine which adds 2milsec wait

Error	Addr	Code	Seq	Source statement
0502	D3 33	1402	V12RET: OUT	CLRACL ;(V12 Added label) Negate ACLO
0504	DB 02	1403	2\$: IN	DCLO ;Monitor DCLO while waiting for ACLO to
0506	17	1404	RAL	;go away.
0507	DA 04EE	1405	JC	1\$;Go try again if DCLO becomes asserted.
050A	20	1406	RIM	;Check ACLO.
050B	17	1407	RAL	;Check the SID (which is ACLO)
050C	DA 0504	1408	JC	2\$;Loop here waiting for ACLO to go away
050F	D3 2E	1409	OUT	CLRBSY ;Negate Bus Busy
0511	C9	1410	RET	;Leave
0512	DB 02	1411		
0514	17	1412	WAIT: IN	DCLO ;Low for DCLO asserted
0515	D8	1413	RAL	;Put it in the carry
0516	0B	1414	RC	;Leave with carry set if DCL set
0517	78	1415	DCX	B ;Decrement the count
0518	B1	1416	MOV	A,B ;Get the high byte of the count
0519	C2 0512	1417	ORA	C ;OR in the low byte
051C	C9	1418	JNZ	WAIT ;Loop if not zero
		1419	RET	;Leave
		1420		
		1421		*****
		1422		
		1423		
		1424		*****
		1425		
		1426		
051D		1427	FLIP_O_FLAG:	
051D	CD 03B1	1428	CALL	FINISH_LOCAL ;Go clear the RD Rcv Done if set
0520	3A 40F2	1429	LDA	SEND_IN_PROGRES ;See if we were in the SEND_CHARACTER routine.
0523	A7	1430	ANA	A ;Is it set?
0524	CA 0531	1431	JZ	1\$;Jump if clear
0527	3A 40F3	1432	LDA	CPFLAG ;Get this flag byte
052A	A7	1433	ANA	A ;Is anything set?
052B	C0	1434	RNZ	;Leave if we are in the middle of ^C, ^P or Brk
052C	E1	1435	POP	H ;Otherwise, fix up the SP
052D	AF	1436	XRA	A ;Make a 0
052E	32 40F2	1437	STA	SEND_IN_PROGRES ;and clear this flag since we aren't going back
0531	21 40BB	1438	1\$: LXI	H,O_FLAG ;Point to the O Flag
0534	7E	1439	MOV	A,M ;Get it
0535	36 00	1440	MVI	M,0 ;Zero out the O Flag
0537	EE 01	1441	XRI	1 ;Flip it
0539	23	1442	INX	H ;Point to the temp storage for the copy
053A	77	1443	MOV	M,A ;Save it
053B	CA 0544	1444	JZ	2\$;Jump if O Flag is now 0
053E	3E 5E	1445	MVI	A,'^' ;Get the first letter to be printed
0540	DF	1446	RST	3 ;Go to SEND_CHARACTER
0541	3E 4F	1447	MVI	A,'O' ;Now the O
0543	DF	1448	RST	3 ;And go print it
0544	3A 40BC	1449	2\$: LDA	O_FLAG_COPY ;Get the real O Flag
0547	32 40BB	1450	STA	O_FLAG ;Store it

Error	Addr	Code	Seq	Source statement
	054A	AF	1451	XRA A ;Clear the carry
	054B	C9	1452	RET ;Leave
			1453	;
			1454	;.....
			1455	;
			1456	;
			1457	;.....
			1458	;
054C	3E 01		1459	CNTLC: MVI A,1 ;Make a 1
054E	32 40F3		1460	STA CPFLAG ;Set this flag for later in CNTLC
0551	21 06C6		1461	LXI H,CNTLC ;Point to this string
0554	22 40F0		1462	CNTLC0: SHLD MSGBUF ;Save the address of the string to be printed
0557	CD 03B1		1463	CALL FINISH_LOCAL ;Go clear the RD Rcv Done if set
055A	3A 408B		1464	LDA CODE_FLAG ;Get the Code flag
055D	21 4080		1465	LXI H,RAM_VALID ;Point to the RAM Valid flag
0560	A6		1466	ANA M ;AND them together
0561	21 4080		1467	LXI H,STACK ;Set up HL with this value
0564	CA 056A		1468	JZ 1\$;IF (CODE_FLAG * RAM_VALID)=0, THEN jump.
0567	2A 4086		1469	LHLD SPBUFF ;ELSE, use this for the SP
056A	F9		1470	1\$: SPHL ;Load the SP
056B	CD 03C7		1471	CALL CTLPRP ;Go share some cleanup code
056E	2A 40F0		1472	LHLD MSGBUF ;Get the address of the string to be printed
0571	CD 059A		1473	CALL FORCE ;Go print it
0574	21 40F3		1474	LXI H,CPFLAG ;Point to this flag byte
0577	7E		1475	MOV A,M ;Get it
0578	1F		1476	RAR ;Put the LSB in the carry. (Asserted=^C)
0579	36 00		1477	MVI M,0 ;Clear all flags
057B	3A 4080		1478	LDA RAM_VALID ;Get the Ram Valid flag into A
057E	D2 0595		1479	JNC 2\$;Jump if C Flag not set
0581	1F		1480	RAR ;Put the Ram Valid Flag in the carry
0582	D2 0160		1481	JNC INIT1 ;Jump if not set
0585	21 409B		1482	LXI H,'JDFLG ;Point to this flag
0588	7E		1483	MOV A,M ;Get it
0589	23		1484	INX H ;Point to the Dir Flag
058A	B6		1485	ORA M ;OR it in
058B	CA 4C28		1486	JZ CVECTOR ;Jump if neither set
058E	AF		1487	XRA A ;Make a zero
058F	77		1488	MOV M,A ;Zero the DIR flag
0590	2B		1489	DCX H ;Point to the Load Flag
0591	77		1490	MOV M,A ;Zero it
0592	C3 4C2B		1491	JMP PVECTOR ;Go through this vector
			1492	;
0595	1F		1493	2\$: RAR ;Put the Ram Valid flag in the carry
0596	DA 4C2B		1494	JC PVECTOR ;Jump if Ram Valid
0599	CF		1495	RST 1 ;Otherwise, stay in the ROM
			1496	;
			1497	;.....
			1498	;
			1499	;
			1500	;.....

Error	Addr	Code	Seq	Source statement
			1501	;
			1502	;
059A	AF		1503	FORCE: XRA A ;Make a 0
059B	32 40BB		1504	STA 0_FLAG ;Zero out this flag
059E	32 40F2		1505	STA SEND_IN_PROGRES ;and this
05A1	32 40CD		1506	STA S_FLAG ;and this one also
05A4	32 40CE		1507	STA DELETE_FLAG ;and another
05A7	F7		1508	RST 6 ;Go print
05A8	3E 0D		1509	MVI A,CR ;Set up a Carriage Return
05AA	DF		1510	RST 3 ;Go print it
05AB	C9		1511	RET ;Leave
			1512	;
			1513	;
			1514	;
			1515	;
			1516	;
			1517	;
05AC	21 06C9		1518	CNTLP: LXI H,CNTLP ;Point to the Control P message
05AF	3E 02		1519	CNTLP0: MVI A,2 ;Make a 2
05B1	32 40F3		1520	STA CPFLAG ;Clear the ^C flag and set the (^P+Brk) flag
05B4	C3 0554		1521	JMP CNTLC0 ;Go share some code
			1522	;
			1523	;
			1524	;
			1525	;
			1526	;
			1527	;
05B7			1528	SEND_LINE:
05B7	7E		1529	MOV A,M ;Get the count
05B8	23		1530	INX H ;Bump the pointer
05B9			1531	SEND_LI_0:
05B9	32 40C3		1532	STA PRICNT ;Save the count
05BC	22 40C1		1533	SHLD PRIADR ;Save the address
05BF	AF		1534	XRA A ;Make a 0
05C0	32 4082		1535	STA BYTSUM ;and zero out the Byte Checksum area
05C3	2A 40C1		1536	1\$: LHLD PRIADR ;Get the address of the line to be printed
05C6	7E		1537	MOV A,M ;Get the character
05C7	23		1538	INX H ;Bump the address along
05C8	22 40C1		1539	SHLD PRIADR ;Save the new value
05CB	DF		1540	RST 3 ;Go send a character
05CC	21 40C3		1541	LXI H,PRICNT ;Point to the count
05CF	35		1542	DCR M ;Subtract one
05D0	C2 05C3		1543	JNZ 1\$;Go send another if not 0
05D3	C9		1544	RET ;Otherwise, leave with Z bit set
			1545	;
			1546	;
			1547	;
			1548	;
			1549	;
			1550	;

Error	Addr	Code	Seq	Source statement
	05D4		1551	SEND_CHARACTER:
	05D4	32 40C0	1552	STA SAVE_CHARACTER ;Save the character
	05D7	3A 40BE	1553	LDA LR_MASK ;Get the Local/Remote mask
	05DA		1554	SEND_CHAO:
	05DA	32 40BF	1555	STA LR_MASK_COPY ;Save a copy of the Mask.
	05DD	3E 01	1556	MVI A,1 ;Make a 1.
	05DF	32 40F2	1557	STA SEND_IN_PROGRES ;Flag to indicate we are in SEND_CHARACTER.
	05E2	3A 40BF	1558	1\$: LDA LR_MASK_COPY ;Get the copy
	05E5	32 40BD	1559	2\$: STA LR_MASK_WORK ;Set up the working version
	05E8	E7	1560	3\$: RST 4 ;Go look for Control C or P
	05E9	3A 7E46	1561	LDA MSG15_COUNT ;Get the Message Type 15 count
	05EC	21 7E49	1562	LXI H,CLMSG_COUNT ;Point to the Connection Lost Msg Count.
	05EF	B6	1563	ORA M ;OR them together.
	05F0	C2 05E8	1564	JNZ 3\$;IF MSG15_COUNT<>0 OR CLMSG_COUNT<>0,
			1565	;THEN loop back.
	05F3	3A 7E04	1566	LDA TALK_FLAG ;ELSE, get the Talk Flag
	05F6	1F	1567	RAR ;Is it set?
	05F7	D2 0600	1568	JNC 4\$;IF NOT SET, THEN skip around next part.
	05FA	CD 080F	1569	CALL TALK_VECTOR ;Go to the Talk flow
	05FD	C3 05E8	1570	JMP 3\$;Loop here
	0600	3A 40CD	1571	4\$: LDA S_FLAG ;Get this flag
	0603	1F	1572	RAR ;Is it set?
	0604	DA 05E8	1573	JC 3\$;Loop here if it is
	0607	3A 40BB	1574	LDA O_FLAG ;Get the O Flag
	060A	1F	1575	RAR ;Is it set?
	060B	D8	1576	RC ;Leave if it is
	060C	21 40BD	1577	LXI H,LR_MASK_WORK ;Point to the working copy
	060F	7E	1578	MOV A,M ;Get it
	0610	E6 07	1579	ANI 7 ;Save only the XMIT bits
	0612	CA 066A	1580	JZ 10\$;Jump if nobody left to be sent to
	0615	1F	1581	RAR ;Check for Local
	0616	D2 0628	1582	JNC 5\$;Jump if not set
	0619	DB 49	1583	IN TTSR ;Get the Local SR
	061B	1F	1584	RAR ;Look for XMIT Ready
	061C	D2 0628	1585	JNC 5\$;Jump if not set
	061F	3A 40C0	1586	LDA SAVE_CHARACTER ;Get the character to be sent
	0622	D3 48	1587	OUT TTDB ;Send it out
	0624	3E FE	1588	MVI A,0FEH ;Get this mask
	0626	A6	1589	ANA M ;Clear the Local Bit
	0627	77	1590	MOV M,A ;Save the new masked version
	0628	7E	1591	5\$: MOV A,M ;Get the Working Copy of the mask
	0629	E6 02	1592	ANI 2 ;See if APT set
	062B	CA 063F	1593	JZ 7\$;Jump if not set
	062E	DB 41	1594	IN TRSR ;Get the remote SR
	0630	1F	1595	RAR ;Look for XMIT RDY
	0631	D2 05E8	1596	JNC 3\$;Jump if not
	0634	3A 40C0	1597	LDA SAVE_CHARACTER ;Get the character to be sent
	0637	D3 40	1598	OUT TRDB ;Send it out
	0639	3E FD	1599	MVI A,0FDH ;Get ready to clear the next bit
	063B	A6	1600	6\$: ANA M ;Do it

Error Addr	Code	Seq	Source statement
063C	C3 05E5	1601	JMP 2\$;Loop back
		1602	
063F	7E	1603	7\$: MOV A,M ;Get the Working Copy of the mask
0640	E6 04	1604	ANI 4 ;See if RD set
0642	CA 05E8	1605	JZ 3\$;Loop back if not set
0645	3A 7E10	1606	LDA RDCON ;See if we are connected
0648	1F	1607	RAR ;Put the flag in the carry
0649	DA 0653	1608	JC 8\$;Jump if we are
064C	3A 7E05	1609	LDA DISCARD_RO ;Get the Discard Remote Output bit
064F	1F	1610	RAR ;Is it set?
0650	DA 0662	1611	JC 9\$;Jump if it is
0653	21 7E02	1612	8\$: LXI H,REMSR ;Point to the pseudo Xmit SR
0656	7E	1613	MOV A,M ;Get it
0657	1F	1614	RAR ;Is it set?
0658	D2 05E8	1615	JNC 3\$;Loop back if not
065B	36 00	1616	MVI M,0 ;Clear it
065D	23	1617	INX H ;Point to the Pseudo DB
065E	3A 40C0	1618	LDA SAVE_CHARACTER ;Get the character to be sent
0661	77	1619	MOV M,A ;Store it
0662	3E FB	1620	9\$: MVI A,0FBH ;Prepare to clear the Remote bit
0664	21 40BD	1621	LXI H,LR_MASK_WORK ;Point to the Working Mask
0667	C3 063B	1622	JMP 6\$;Go share some code
		1623	
		1624	
066A	3A 40C0	1625	10\$: LDA SAVE_CHARACTER ;Get the character
066D	47	1626	MOV B,A ;Save a copy
066E	21 4082	1627	LXI H,BYTSUM ;Point to the byte checksum area
0671	86	1628	ADD M ;Add it in
0672	77	1629	MOV M,A ;Put it back
0673	3A 40FF	1630	LDA NOCHK ;See if we should check for CR
0676	1F	1631	RAR ;Is checking inhibited?
0677	DA 0688	1632	JC 11\$;Leave if it is
067A	78	1633	MOV A,B ;Get the copy in B
067B	FE 0D	1634	CPI CR ;See if it was a Carriage Return
067D	C2 0688	1635	JNZ 11\$;Leave if it wasn't
0680	3E 0A	1636	MVI A,LF ;Put a Line Feed in A
0682	32 40C0	1637	STA SAVE_CHARACTER ;Put it in the storage area
0685	C3 05E2	1638	JMP 1\$;Go back and do it again
0688	AF	1639	11\$: XRA A ;Make a 0
0689	32 40F2	1640	STA SEND_IN_PROGRES ;Zero this flag
068C	C9	1641	RET ;Leave
		1642	;
		1643	;*****
		1644	
		1645	
		1646	
		1647	
		1648	;*****
		1649	;
068D		1650	MODEM_ENTRY:

Error	Addr	Code	Seq	Source statement
068D	2A	4099	1651	LHLD MODEM_ADDRESS ;Get the entry point address
0690	E9		1652	PCHL ;and go there.
			1653	
0691	CD	1004	1654	FIXUP: CALL FA_VECTOR ;Point to MODEM0
0694	DB	06	1655	MODEM0: IN REMOTE ;Check the switch position
0696	1F		1656	RAR ;Put it in carry for checking
0697	D8		1657	RC ;Leave if in Local
0698	DB	04	1658	IN APTFLG ;Check for APT
069A	1F		1659	RAR ;Put in carry
069B	D0		1660	RNC ;Leave if APT present
069C	3A	10C0	1661	LDA RD_PATTERN ;Get the first byte from the next set of ROM
069F	FE	A5	1662	CPI 0A5H ;Check for the correct pattern
06A1	C0		1663	RNZ ;Leave if not
06A2	C3	1007	1664	JMP MOD0_VECTOR ;Jump into next part of ROM
			1665	
			1666	;
			1667	;*****
			1668	
			1669	
			1670	;*****
			1671	;
			1672	;This is the Diagnostic Print routine. The 'caller' must have HL pointing
			1673	;to the character to be printed and the return point is assumed to be the
			1674	;address following this character.
			1675	;
06A5			1676	DPRINT:
06A5	DB	49	1677	1\$: IN TTSR ;Check the XMIT Ready bit
06A7	1F		1678	RAR ;Put it in the carry
06A8	D2	06A5	1679	JNC 1\$;Loop back if not ready
06AB	7E		1680	MOV A,M ;Get the character to be printed
06AC	D3	48	1681	OUT TTDB ;Send it to the Local Terminal
06AE	23		1682	INX H ;Bump the pointer
06AF	DB	49	1683	2\$: IN TTSR ;Wait for ready to come back before leaving
06B1	1F		1684	RAR ;
06B2	D2	06AF	1685	JNC 2\$;Loop until Xmit Ready is asserted again
06B5	E9		1686	PCHL ;Leave
			1687	;
			1688	;*****
			1689	
			1690	
			1691	;*****
			1692	;
06B6	04	04	1693	INITPK: DB TUINIT,TUINIT ;TU-58 Init code
			1694	;
			1695	;*****
			1696	
			1697	
			1698	;*****
			1699	;
06B8	02	0A 02	1700	BOOTPK: DB 02,0AH,02,0

Error	Addr	Code	Seq	Source statement
06BB	00			
06BC	01 00		1701	DB 1,0 ;
06BE	0000		1702	DW 0 ;
06C0	0C00		1703	DW 3072 ;
06C2	0000		1704	DW 0 ;
06C4	0694		1705	VECTOR: DW MODEMO ;Address into Modem Control routines
			1706	;
			1707	*****
			1708	
			1709	
			1710	*****
			1711	;
06C6	02 5E 43		1712	CONTLC: DB 2,'^C' ;
06C9	02 5E 50		1713	CONTLP: DB 2,'^P' ;
06CC	01 00		1714	BRKMSG: DB 1,0 ;
			1715	;
			1716	*****
			1717	
			1718	
			1719	*****
			1720	;
			1721	These are the tape error messages
			1722	;
06CE	12 0D 3F		1723	DEVERR: DB 18,CR,'?27 DEVICE ERROR' ;Init or Timeout Errors.
06D1	32 37 20			
06D4	20 44 45			
06D7	56 49 43			
06DA	45 20 45			
06DD	52 52 4F			
06E0	52			
			1724	
06E1	10 0D 3F		1725	REDERR: DB 16,CR,'?27 READ ERROR' ;Tape Checksum Errors.
06E4	32 37 20			
06E7	20 52 45			
06EA	41 44 20			
06ED	45 52 52			
06F0	4F 52			
			1726	
06F2	0D 0D 3F		1727	NOBOOT: DB 13,CR,'?40 NO BOOT' ;
06F5	34 30 20			
06F8	20 4E 4F			
06FB	20 42 4F			
06FE	4F 54			
			1728	
0700	03 20 44		1729	DRVMSG: DB 3,' DD ;
0703	44			
			1730	;
			1731	End of Tape Error messages
			1732	;
			1733	*****

Error	Addr	Code	Seq	Source statement
-------	------	------	-----	------------------

		1734	
		1735	
		1736	;*****
		1737	;
		1738	; Other messages
		1739	;
0704		1740	NACK_X_CMND:
0704	03 3F 32	1741	SYNTAX: DB 3,'?20' ;
0707	30		
		1742	
0708	05 0D 52	1743	RPRMPT: DB 5,CR,'ROM>' ;
070B	4F 4D 3E		
		1744	
070E		1745	NACK_X_DATA:
070E	04 0D 3F	1746	DB 4,CR,'?30'
0711	33 30		
		1747	
0713		1748	VALIDITY_ERR:
0713	04 0D 3F	1749	DB 4,CR,'?40'
0716	34 30		
		1750	;
		1751	; End of other messages
		1752	;
		1753	;*****

**	0718	D3 33	V12FIX: OUT	CLRACL	;V12 Negate AC low
**	071A	01 014E	LXI	B,334	;V12 Set up 2 milsec count
**	071D	0B	1\$: DCX	B	;V12 Count it down
**	071E	79	MOV	A,C	;V12 Get low byte into A
**	071F	B0	ORA	B	;V12 OR in high byte
**	0720	C2 071D	JNZ	1\$	;V12 Jump if still time
**	0723	3E 10	MVI	A,010H	;V12 Reset RST 7.5
**	0725	30	SIM		;V12 Do it
**	0726	C3 0502	JMP	V12RET	;V12 Return to flow

```

1754
1755
07FF  =07FF 1756          ORG      7FFH          ;
1757
1758 LOWSUM: DB      0          ;
1759
1760
1761
1762
1763 ;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
1764 ;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
1765 ;!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
1766 ;
1767 ;
1768
1769
      =0800 1770          ORG      800H          ;
1771

```

0800		1772	RD_LD_VECTOR:		
0800	C3 0B70	1773	JMP	RD_LD_COMMAND	;Go to the RD LOAD Command flows
		1774			
0803		1775	DCD_VECTOR:		
0803	C3 0BE1	1776	JMP	DCD_MESSAGE	;Go the (Diana) Control Data Message flows
		1777			
0806		1778	TEXT_VECTOR:		
0806	C3 0CE2	1779	JMP	TEXT_MESSAGE	;Go to the Text Message flows

Error	Addr	Code	Seq	Source statement
			1780	
0809			1781	STT_VECTOR:
0809	C3 0B25		1782	JMP SEND_TRANS_TEXT ;
			1783	
080C			1784	RL_VECTOR:
080C	C3 0A6A		1785	JMP RAM_LOAD ;Go to Ram Load.
			1786	
080F			1787	TALK_VECTOR:
080F	C3 0CFB		1788	JMP TALK ;Go to the Talk Mode flows.
			1789	
0812			1790	MD_VECTOR:
0812	C3 0D88		1791	JMP MSG_DECODE ;Go to Message Decode
			1792	
0815			1793	ST_VECTOR:
0815	C3 082A		1794	JMP SELF_TEST ;Go to the Self Test.
			1795	
0818			1796	BUILD_VECTOR:
0818	C3 0E2E		1797	JMP BUILD ;Go to the Build (the Out Packet) flow.
			1798	
081B			1799	BP_VECTOR:
081B	C3 0E67		1800	JMP BUILD_PREP ;Go to Build_Prep.
			1801	
081E	C3 081E		1802	MRVEC8: JMP MRVEC8 ;TEMPORARY
			1803	
0821	C3 0821		1804	MRVEC9: JMP MRVEC9 ;TEMPORARY
			1805	
0824	C3 0824		1806	MRVE10: JMP MRVE10 ;TEMPORARY
			1807	
0827	C3 0827		1808	MRVE11: JMP MRVE11 ;TEMPORARY
			1809	
			1810	
			1811	;.....
			1812	; ROM - NEBULA SELF TESTS
			1813	;.....
			1814	
			1815	
			1816	
082A			1817	SELF_TEST:
082A	D3 35		1818	OUT INITH ;Set Unibus Init
082C	D3 2F		1819	OUT SETBSY ;Set Bus Busy
082E	DB 07		1820	1\$: IN OK15V ;Check the + and - 15 volts
0830	17		1821	RAL ;The flag is in the MSB
0831	D2 082E		1822	JNC 1\$;Loop until set
0834	DB 47		1823	IN TUCR ;These IN's force an internal pointer
0836	DB 4B		1824	IN TTCR ;in the USARTs to point to the MODE1
0838	DB 43		1825	IN TRCR ;register
083A	3E 4E		1826	MVI A,4EH ;1 Stop bit,8 bits wide,ASYNC,Divide by 16
083C	D3 46		1827	OUT TUMODE ;Send it out to the USARTs
083E	D3 4A		1828	OUT TTMODE
0840	D3 42		1829	OUT TRMODE

Error	Addr	Code	Seq	Source statement
	0842	3E 35	1830	MVI A,TTCR_CODE ;RTS (DSRS for Europe RD), Error Reset, Xmit
			1831	;and Rcv Enable
	0844	D3 4B	1832	OUT TTCR ;Set up the terminal USART
	0846	DB 04	1833	IN READJ2 ;Check the status of J2
	0848	17	1834	RAL ;Put it in the carry
	0849	D2 0024	1835	JNC INIT_VECTOR ;Jump if low
			1836	
			1837	
			1838	;Since subroutines can't be used yet, the print routine is shared in the
			1839	;following manner:
			1840	:
			1841	: 1) The 'caller' must load HL with the address of the character
			1842	to be printed.
			1843	: 2) The return point immediately follows the character.
			1844	: 3) An RST 7 is used to get to the DPRINT routine. Since the SP
			1845	gets Pushed twice, the state of the LSB is preserved.
			1846	:
			1847	
	084C	21 0850	1848	LXI H,FIRST_CHAR ;Point to the LF
	084F	FF	1849	RST 7 ;Go print it
	0850		1850	FIRST_CHAR:
	0850	0A	1851	DB LINE_FEED ;Line Feed code.
	0851	21 0855	1852	LXI H,SECOND_CHAR ;Point to the CR
	0854	FF	1853	RST 7 ;Go print it
	0855		1854	SECOND_CHAR:
	0855	0D	1855	DB CAR_RET ;Carriage Return code.
	0856	21 085A	1856	LXI H,THIRD_CHAR ;Point to this return point
	0859	FF	1857	RST 7 ;Go to the Diagnostic print routine
	085A		1858	THIRD_CHAR:
	085A	43	1859	DB C_LETTER ;This is the letter to print
			1860	
			1861	
			1862	
			1863	
			1864	:
			1865	:
			1866	: BEGINNING OF ROM CHECKSUM
			1867	:
	085B		1868	ROM_CHKSUM:
	085B	06 00	1869	MVI B,0 ;Clear out this flag
	085D	D3 A8	1870	OUT SETACK ;Set the Scope Sync signal
	085F	D3 A9	1871	OUT CLRACK ;Clear it
	0861	21 0000	1872	LXI H,0 ;Clear HL because ROM starts at 0
	0864	54	1873	MOV D,H ;Clear D
			1874	
			1875	
			1876	:
			1877	: This is where we sum up the first 2 kbyte segment (ROMS E53 and E54).
			1878	:
	0865	7A	1879	1\$: MOV A,D ;Get accumulated checksum value

Error	Addr	Code	Seq	Source statement
	0866	86	1980	ADD M ;Add contents of current memory location
	0867	57	1881	MOV D,A ;Save result
	0868	23	1882	INX H ;Bump the pointer
	0869	3E 08	1883	MVI A,8H ;Check for the end of the first 2k
	086B	BC	1884	CMP H ;We only need to look at the high address byte
	086C	C2 0865	1885	JNZ 1\$;Keep adding them up
	086F	7A	1886	MOV A,D ;Get the result
	0870	A7	1887	ANA A ;Check for 0
	0871	C2 0871	1888	JNZ 2\$;Stay here if not
			1889	;
			1890	;
			1891	;
			1892	;
			1893	;
			1894	;
			1895	Now we do the second 2 kbyte segment (ROMS E49 and E51).
	0874	7A	1896	3\$: MOV A,D ;Get accumulated checksum value
	0875	86	1897	ADD M ;Add contents of current memory location
	0876	57	1898	MOV D,A ;Save result
	0877	23	1899	INX H ;Bump the pointer
	0878	3E 10	1900	MVI A,10H ;Check for the end of the second 2k
	087A	BC	1901	CMP H ;We only need to look at the high address byte
	087B	C2 0874	1902	JNZ 3\$;Keep adding them up
	087E	7A	1903	MOV A,D ;Get the result
	087F	A7	1904	ANA A ;Check for 0
	0880	C2 0880	1905	JNZ 4\$;Stay here if not
			1906	;
			1907	;
			1908	;
			1909	;
			1910	;
			1911	;
			1912	;
			1913	Last, see if the high 2 kbyte segment is installed (ROMS E50 and E52).
			1914	If not, we skip this part. If it is, we checksum it
	0883	7E	1915	MOV A,M ;Get the data at 1000H
	0884	FE A5	1916	CPI 0A5H ;0A5H indicates last 2k is there
	0886	C2 0898	1917	JNZ PRINT_FOURTH ;Skip the next part if no ROM starting at 1000H
	0889	7A	1918	5\$: MOV A,D ;Get accumulated checksum value
	088A	86	1919	ADD M ;Add contents of current memory location
	088B	57	1920	MOV D,A ;Save result
	088C	23	1921	INX H ;Bump the pointer
	088D	3E 18	1922	MVI A,18H ;Check for the end of the third 2k
	088F	BC	1923	CMP H ;We only need to look at the high address byte
	0890	C2 0889	1924	JNZ 5\$;Keep adding them up
	0893	7A	1925	MOV A,D ;Get the result
	0894	A7	1926	ANA A ;Check for 0
	0895	C2 0895	1927	JNZ 6\$;Stay here if not
			1928	;
			1929	Otherwise, fall through to next test

Error	Addr	Code	Seq	Source statement
			1930	*****
			1931	;
			1932	END OF ROM CHECKSUM
			1933	;
			1934	*****
			1935	*****
			1936	;
			1937	*****
			1938	;
			1939	;
0898			1940	PRINT_FOURTH:
0898	21	089C	1941	LXI H,FOURTH_CHAR ;Point to this character
089B	FF		1942	RST 7 ;Go print the character
089C			1943	FOURTH_CHAR:
089C	4F		1944	DB 0_LETTER ;This is the character to be printed
089D	21	08A1	1945	LXI H,FIFTH_CHAR ;Point to a null
08A0	FF		1946	RST 7 ;Go print it
08A1			1947	FIFTH_CHAR:
08A1	00		1948	DB 0 ;Null
08A2	AF		1949	XRA A ;Make a 0.
08A3	D3	4B	1950	OUT TTCR ;This is for the 2661 USARTS.
08A5	D3	47	1951	OUT TUCR
08A7	D3	43	1952	OUT TRCR
08A9	3E	B3	1953	MVI A,0B3H ;Now we can go into Local Loop Back.
08AB	D3	4B	1954	OUT TTCR
08AD	D3	47	1955	OUT TUCR
08AF	D3	43	1956	OUT TRCR
			1957	*****
			1958	*****
			1959	*****
			1960	;
			1961	BEGINNING OF LOCAL LOOPBACK USART TEST
			1962	;
			1963	By using a -1 in E and XORing the pattern with it, we generate
			1964	a 0 in a background of ones. After rippling through this, we bump
			1965	the value in E to 0 and now the XOR does nothing (effectively) and
			1966	we ripple through ones with a background of zeros. The value in E
			1967	also acts as loop control.
			1968	;
			1969	*****
			1970	*****
			1971	;
			1972	Set up the patterns
			1973	;
08B1	1E	FF	1974	MVI E,0FFH ;Put a minus one in E
08B3	21	0102	1975	LXI H,102H ;H gets 1 and L gets 2
			1976	;
			1977	*****
			1978	*****
			1979	*****

Error Addr	Code	Seq	Source statement
		1980	*****
		1981	;
08B6		1982	LOCAL_LOOPBACK:
08B6	7C	1983	MOV A,H ;Get the pattern
08B7	AB	1984	XRA E ;XOR it against the value in E
08B8	57	1985	MOV D,A ;Put this in reg D
		1986	*****
		1987	;
		1988	;
		1989	Test USART 1
		1990	;
08B9		1991	USART1_LL:
08B9	D3 A8	1992	OUT SETACK ;Set the Scope Sync
08BB	D3 A9	1993	OUT CLRACK ;Clear it
08BD	DB 49	1994	1\$: IN TTSR ;Wait for XMIT RDY
08BF	1F	1995	RAR
08C0	D2 08BD	1996	JNC 1\$
08C3	7A	1997	MOV A,D ;Get the pattern
08C4	D3 48	1998	OUT TTDB ;Send it out
08C6	DB 49	1999	2\$: IN TTSR ;Get the SR
08C8	A5	2000	ANA L ;Check for Rcvr Done
08C9	CA 08C6	2001	JZ 2\$;Wait for it
08CC	DB 48	2002	IN TTDB ;Get the data
08CE	BA	2003	CMP D ;Check it
08CF	CA 08D4	2004	JZ 3\$;Jump if OK
08D2	06 01	2005	MVI B,1 ;Set the error flag
08D4	78	2006	3\$: MOV A,B ;Get the error flag
08D5	1F	2007	RAR ;Is it set?
08D6	DA 08B9	2008	JC USART1_LL ;Jump if it is
		2009	;
		2010	*****
		2011	;
		2012	;
		2013	*****
		2014	;
		2015	Test USART 2
		2016	;
08D9		2017	USART2_LL:
08D9	D3 A8	2018	OUT SETACK ;Set the Scope Sync
08DB	D3 A9	2019	OUT CLRACK ;Clear it
08DD	DB 45	2020	1\$: IN TUSR ;Wait for XMIT RDY
08DF	1F	2021	RAR
08E0	D2 08DD	2022	JNC 1\$
08E3	7A	2023	MOV A,D ;Get the data to be sent
08E4	D3 44	2024	OUT TUDB ;Send it
08E6	DB 45	2025	2\$: IN TUSR ;Get the SR
08E8	A5	2026	ANA L ;Check for Rcvr Done
08E9	CA 08E6	2027	JZ 2\$;Loop until it is
08EC	DB 44	2028	IN TUDB ;Get the character
08EE	BA	2029	CMP D ;Is it correct?

Error	Addr	Code	Seq	Source statement
	08EF	CA 08F4	2030	JZ 3\$;Jump if OK
	08F2	06 01	2031	MVI B,1 ;Set the error flag
	08F4	78	2032	3\$: MOV A,B ;Get the error flag
	08F5	1F	2033	RAR ;Is it set?
	08F6	DA 08D9	2034	JC USART2_LL ;Jump if it is
			2035	;
			2036	;.....
			2037	;
			2038	;
			2039	;.....
			2040	;
			2041	; Test USART 3
			2042	;
	08F9		2043	USART3_LL:
	08F9	D3 A8	2044	OUT SETACK ;Set the Scope Sync
	08FB	D3 A9	2045	OUT CLRACK ;Clear it
	08FD	DB 41	2046	1\$: IN TRSR ;Get the SR for the third USART
	08FF	1F	2047	RAR ;Look for Xmit Rdy
	0900	D2 08FD	2048	JNC 1\$;Loop if not Ready
	0903	7A	2049	MOV A,D ;Get the data to be sent
	0904	D3 40	2050	OUT TRDB ;Send it
	0906	DB 41	2051	2\$: IN TRSR ;Get the SR again
	0908	A5	2052	ANA L ;Check for Rcvr Done
	0909	CA 0906	2053	JZ 2\$;Wait for it
	090C	DB 40	2054	IN TRDB ;Get the character
	090E	BA	2055	CMP D ;Check it
	090F	CA 0914	2056	JZ 3\$;Jump if OK
	0912	06 01	2057	MVI B,1 ;Set the error flag
	0914	78	2058	3\$: MOV A,B ;Get the error flag
	0915	1F	2059	RAR ;Is it set?
	0916	DA 08F9	2060	JC USART3_LL ;Jump if it is
			2061	;
			2062	;.....
			2063	;
			2064	;
			2065	;.....
			2066	;
			2067	; Now rotate the pattern and see if we are done. If not, jump back to
			2068	;
	0919	7C	2069	MOV A,H ;Get the pattern
	091A	07	2070	RLC ;Rotate left one place
	091B	67	2071	MOV H,A ;Make this the new pattern
	091C	D2 08B6	2072	JNC LOCAL_LOOPBACK ;Keep looping until the one shifts out
	091F	1C	2073	INR E ;Bump the value in E
	0920	CA 08B6	2074	JZ LOCAL_LOOPBACK ;Loop back until not 0
	0923	AF	2075	XRA A ;GET A 0
	0924	D3 4B	2076	OUT TTCR ;FOR 2661
	0926	D3 47	2077	OUT TUCR ;
	0928	D3 43	2078	OUT TRCR ;
			2079	;

Error	Addr	Code	Seq	Source statement
			2080	;*****
			2081	
			2082	
			2083	;*****
			2084	;
092A	3E 15		2085	MVI A,15H ;Code for Error Reset, Xmit and Rcv Enables
092C	D3 4B		2086	OUT TTCR ;Send it out to the Terminal CR
			2087	;and fall through to the next test
			2088	;
			2089	END OF USART LOCAL LOOPBACK TEST
			2090	;
			2091	;*****
			2092	;*****
			2093	
			2094	
			2095	
			2096	
			2097	;*****
			2098	;
			2099	UBTST
			2100	;
			2101	;The second USART test is a dual addressing test. A different value is
			2102	;written to each of the Command Registers and then read back. If the value
			2103	;read back is incorrect, a dual addressing problem is assumed.
			2104	;
			2105	;*****
			2106	
			2107	
092E	21 0932		2108	UBTST: LXI H,UBX ;Point to this character
0931	FF		2109	RST 7 ;Go print it
0932	4E		2110	UBX: DB N_LETTER ;This is the character
0933	11 0104		2111	LXI D,104H ;Put a 1 in D and a 4 in E
			2112	
0936	D3 A8		2113	UB1: OUT SETACK ;Set the Scope Sync
0938	D3 A9		2114	OUT CLRACK ;Clear it
093A	7A		2115	MOV A,D ;Pattern=1 for USART 1
093B	D3 4B		2116	OUT TTCR ;Send it to the Command Register
093D	7B		2117	MOV A,E ;Pattern=4 for USART 2
093E	D3 47		2118	OUT TUCR ;Send it to the CR
0940	3C		2119	INR A ;Pattern=5 for USART 3
0941	D3 43		2120	OUT TRCR ;Send it to the CR
0943	DB 4B		2121	IN TTCR ;Get the Terminal CR back
0945	BA		2122	CMP D ;Is it OK?
0946	CA 094B		2123	JZ UB1A ;Jump if good
0949	06 01		2124	MVI B,1 ;Set the error flag
094B	78		2125	UB1A: MOV A,B ;Get the error flag
094C	1F		2126	RAR ;Put it in the carry
094D	DA 0936		2127	JC UB1 ;Jump if set
			2128	
0950	D3 A8		2129	UB2: OUT SETACK ;Set the Scope Sync

Error	Addr	Code	Seq	Source statement
0952	D3 A9	2130		OUT CLRACK ;Clear it
0954	7A	2131		MOV A,D ;Pattern=1 for USART 2
0955	D3 47	2132		OUT TUCR ;Send it to the Command Register
0957	7B	2133		MOV A,E ;Pattern=4 for USART 1
0958	D3 4B	2134		OUT TTCR ;Send it to the CR
095A	3C	2135		INR A ;Pattern=5 for USART 3
095B	D3 43	2136		OUT TRCR ;Send it to USART3
095D	DB 47	2137		IN TUCR ;Get the USART 2 CR
095F	BA	2138		CMP D ;Is it OK?
0960	CA 0965	2139		JZ UB2A ;Jump if OK
0963	06 01	2140		MVI B,1 ;Set the error flag
0965	78	2141	UB2A:	MOV A,B ;Get the error flag
0966	1F	2142		RAR ;Put it in the carry
0967	DA 0950	2143		JC UB2 ;Jump if set
		2144		
096A	D3 A8	2145	UB3:	OUT SETACK ;Set the Scope Sync
096C	D3 A9	2146		OUT CLRACK ;Clear it
096E	7A	2147		MOV A,D ;Pattern=1 for USART 3
096F	D3 43	2148		OUT TRCR ;Send it to the Command Register
0971	7B	2149		MOV A,E ;Pattern=4 for USART 2
0972	D3 47	2150		OUT TUCR ;Send it to the CR
0974	3C	2151		INR A ;Pattern=5 for USART 1
0975	D3 4B	2152		OUT TTCR ;Send it to USART 1
0977	DB 43	2153		IN TRCR ;Get the USART 3 CR
0979	BA	2154		CMP D ;Is it OK?
097A	CA 097F	2155		JZ UB3A ;Jump if OK
097D	06 01	2156		MVI B,1 ;Set the error flag
097F	78	2157	UB3A:	MOV A,B ;Get the error flag
0980	1F	2158		RAR ;Put it in the carry
0981	DA 096A	2159		JC UB3 ;Jump if set
		2160		
		2161		;We come here if the test completes successfully. The USARTs are fixed
		2162		;and we fall through to the next test.
		2163		
0984	3E 35	2164	UA105:	MVI A,TTCR_CODE ;RTS (DSRS for Europe RD), Error Reset, Xmit
		2165		;and Rcv Enable
0986	D3 4B	2166		OUT TTCR ;
		2167		
		2168		
		2169		;.....
		2170		;
		2171		; RIPPLE
		2172		;
		2173		; THIS TEST WILL RIPPLE A 1 THROUGH A FIELD OF ZEROS IN THE FIRST
		2174		; MEMORY LOCATION (4000) HEX
		2175		;
		2176		;.....
		2177		
0988	21 098C	2178	RIPPLE:	LXI H,RIPPLX ;Point to this character to be printed
098B	FF	2179		RST 7 ;Go print it

Error Addr	Code	Seq	Source statement
098C	56	2180	RIPPLX: DB V_LETTER ;Code for letter V.
098D	16 01	2181	MVI D,1 ;Put the seed pattern in D
098F	21 4000	2182	LXI H,4000H ;Point to the beginning of RAM
0992	4D	2183	MOV C,L ;Put a 0 in C to indicate Ripple Test
0993	72	2184	RIPPL1: MOV M,D ;Write pattern to memory
0994	5E	2185	MOV E,M ;Read it back
0995	78	2186	MOV A,B ;Get the error flag
0996	1F	2187	RAR ;Is it set?
0997	DA 0993	2188	JC RIPPL1 ;Jump if it is
099A	7B	2189	MOV A,E ;Put the data in A
099B	BA	2190	CHP D ;See if it is the same
099C	C2 0A1B	2191	JNZ EPRINT ;Jump if not the same
		2192	
		2193	;Come here to rotate the pattern and check for done.
		2194	;The pattern is already in A
		2195	
099F	07	2196	RLC ;Rotate the pattern to the left
09A0	57	2197	MOV D,A ;Save the new pattern
09A1	D2 0993	2198	JNC RIPPL1 ;Loop until the carry is set
		2199	;Then fall through to next test
		2200	
		2201	
		2202	;
		2203	;
		2204	; MARCH
		2205	;
		2206	; THIS ROUTINE WRITES A BACKGROUND OF ZEROS TO THE RAM AND THEN
		2207	; GOES FROM THE LOWEST TO THE HIGHEST ADDRESS READING THE ZERO BACKGROUND
		2208	; WRITING A 1 AND READING THE 1 BACK. THEN THE TEST GOES FROM THE
		2209	; HIGHEST ADDRESS TO THE LOWEST ADDRESS READING THE 1'S BACKGROUND
		2210	; WRITTING A ZERO AND READING A 0
		2211	;
		2212	;
		2213	;
09A4	21 09A8	2214	MARCH: LXI H,MARCHX ;Point to character to be printed
09A7	FF	2215	RST 7 ;Go do it
09A8	30	2216	MARCHX: DB HIGH_NUMBER ;High digit of version number.
09A9	21 4000	2217	LXI H,4000H ;Point to the beginning of RAM
09AC	AF	2218	MARCH0: XRA A ;Make a 0
09AD	77	2219	MOV M,A ;Write 0's to memory
09AE	23	2220	INX H ;Bump the address pointer
09AF	7C	2221	MOV A,H ;Get the high byte of the address
09B0	17	2222	RAL ;See if the MSB is set
09B1	D2 09AC	2223	JNC MARCH0 ;Jump if not
		2224	;Otherwise, fall through and set up
09B4	EB	2225	XCHG ;a count in DE (count=8000H)
09B5	1B	2226	MARCH1: DCX D ;This loop will wait approximately
09B6	7A	2227	MOV A,D ;40 milliseconds to give the refresh
09B7	B3	2228	ORA E ;logic plenty of exercise.
09B8	C2 09B5	2229	JNZ MARCH1 ;Note that the count is generated by

Error Addr	Code	Seq	Source statement
		2230	;using the 8000(hex) that ends up in
		2231	;HL at the end of writing to memory
		2232	
09BB	21 4000	2233	LXI H,4000H ;Point to the start of the RAM
09BE	0E 01	2234	MARCH2: MVI C,1 ;Set up this Return Point code
09C0	D3 A8	2235	MARCH3: OUT SETACK ;Scope Sync
09C2	D3 A9	2236	OUT CLRACK ;
09C4	5E	2237	MOV E,M ;Get the character
09C5	78	2238	MOV A,B ;Check the Error flag
09C6	1F	2239	RAR ;Is it set?
09C7	DA 09C0	2240	JC MARCH3 ;Loop back if it is
09CA	7B	2241	MOV A,E ;Get the data into A
09CB	BA	2242	CMP D ;See if it is zero
09CC	C2 0A1B	2243	JNZ EPRINT ;Jump if bad
09CF	15	2244	DCR D ;Make -1(Hex) in D
09D0	0C	2245	INR C ;Make the value in C a 2
09D1	D3 A8	2246	MARCH4: OUT SETACK ;Scope Sync
09D3	D3 A9	2247	OUT CLRACK ;
09D5	72	2248	MOV M,D ;Send it to memory
09D6	5E	2249	MOV E,M ;Read it back
09D7	78	2250	MOV A,B ;Get the Error Flag
09D8	1F	2251	RAR ;Is it set?
09D9	DA 09D1	2252	JC MARCH4 ;Loop if it is
09DC	7B	2253	MOV A,E ;Get the data into A
09DD	BA	2254	CMP D ;See if it is OK
09DE	C2 0A1B	2255	JNZ EPRINT ;Jump if bad
09E1	14	2256	INR D ;Make a 0 in D again
09E2	23	2257	INX H ;Bump the address pointer
09E3	7C	2258	MOV A,H ;Get the high byte of the address
09E4	17	2259	RAL ;Check for MSB set
09E5	D2 09BE	2260	JNC MARCH2 ;Loop back if not set
		2261	
		2262	; Now we want to get the 8000H from HL into DE without destroying
		2263	; HL. This is why we don't use XCHG
		2264	
09E8	54	2265	MOV D,H ;Set up Refresh wait count
09E9	5D	2266	MOV E,L ;Count=8000(hex)
09EA	1B	2267	MARCH5: DCX D ;Give the refresh logic 40
09EB	7A	2268	MOV A,D ;milliseconds exercise
09EC	B3	2269	ORA E ;Note that the 8000 in DE comes from
09ED	C2 09EA	2270	JNZ MARCH5 ;HL which has this value in it after writing
		2271	;through memory
		2272	
		2273	; The next test assumes that HL contains 8000(hex) in the
		2274	;beginning.
		2275	
09F0	2B	2276	MARCH6: DCX H ;Bump the address pointer
09F1	7C	2277	MOV A,H ;Look for the bottom address
09F2	FE 3F	2278	CPI 3FH ;
09F4	CA 0ASD	2279	JZ ST_DONE ;Jump out when done

Error	Addr	Code	Seq	Source statement
	09F7	15	2280	DCR D ;Make a -1 in D
	09F8	0E 03	2281	MVI C,3 ;Make the value in C a 3
	09FA	D3 A8	2282	MARCH7: OUT SETACK ;Scope Sync
	09FC	D3 A9	2283	OUT CLRACK ;
	09FE	5E	2284	MOV E,M ;Get what should be a -1
	09FF	78	2285	MOV A,B ;Check the Error Flag
	0A00	1F	2286	RAR ;Is it set?
	0A01	DA 09FA	2287	JC MARCH7 ;Loop if it is
	0A04	7B	2288	MOV A,E ;Put the data in A
	0A05	BA	2289	CMP D ;Check it
	0A06	C2 0A1B	2290	JNZ EPRINT ;Jump if bad
	0A09	14	2291	INR D ;Make a 0 in D
	0A0A	0C	2292	INR C ;Make the value in C a 4
	0A0B	D3 A8	2293	MARCH8: OUT SETACK ;Scope Sync
	0A0D	D3 A9	2294	OUT CLRACK ;
	0A0F	72	2295	MOV M,D ;Send it to memory
	0A10	5E	2296	MOV E,M ;Read it back
	0A11	78	2297	MOV A,B ;Get the Error Flag into A
	0A12	1F	2298	RAR ;Is it set?
	0A13	DA 0A0B	2299	JC MARCH8 ;Loop if it is
	0A16	7B	2300	MOV A,E ;Put the data in A
	0A17	BA	2301	CMP D ;Check it
	0A18	CA 09F0	2302	JZ MARCH6 ;Jump if OK
			2303	;Otherwise, fall through
			2304	
			2305	
			2306	*****
			2307	;
			2308	;
			2309	;
			2310	;
			2311	;
			2312	;
			2313	;
			2314	;
			2315	;
			2316	;
			2317	;
			2318	;
			2319	;
			2320	;
			2321	;
			2322	;
			2323	;
			2324	;
	0A1B	06 10	2325	EPRINT: MVI B,10H ;Set up a count in B for printing
	0A1D	DB 49	2326	EPRIN1: IN TTSR ;Get the terminal SR
	0A1F	1F	2327	RAR ;Check the Xmit Ready bit
	0A20	D2 0A1D	2328	JNC EPRIN1 ;Loop until ready
	0A23	3E 20	2329	MVI A,' ' ;Get a Space

Error	Addr	Code	Seq	Source statement	
0A25	D3 48		2330	OUT TTDB	;Send it out
0A27	DB 49		2331	EPRIN2: IN TTSR	;Get the Status Register
0A29	1F		2332	RAR	;Check for Xmit Ready
0A2A	D2 0A27		2333	JNC EPRIN2	;Loop until it is
0A2D	7A		2334	MOV A,D	;Get the data to be printed
0A2E	07		2335	RLC	;Rotate the MSB into the carry
0A2F	57		2336	MOV D,A	;Put the rotated data back
0A30	3E 30		2337	MVI A,'0'	;Get ASCII 0
0A32	D2 0A36		2338	JNC EPRIN3	;Jump if the MSB was 0
0A35	3C		2339	INR A	;Make an ASCII 1
0A36	D3 48		2340	EPRIN3: OUT TTDB	;Send it out
0A38	05		2341	DCR B	;Reduce the count in B
0A39	78		2342	MOV A,B	;Get the new contents of B into A
0A3A	E6 07		2343	ANI 07H	;Check the low three bits for 0
0A3C	C2 0A27		2344	JNZ EPRIN2	;Loop back if not
0A3F	7A		2345	MOV A,D	;A gets D
0A40	53		2346	MOV D,E	;D gets E
0A41	5F		2347	MOV E,A	;E gets A which has the previous D
0A42	78		2348	MOV A,B	;Get the value in B
0A43	A7		2349	ANA A	;Check for 0
0A44	C2 0A1D		2350	JNZ EPRIN1	;If not zero, go print the second value
0A47	06 01		2351	MVI B,1	;Set up the error code
0A49	79		2352	MOV A,C	;Get the Return Point code
0A4A	FE 01		2353	CPI 1	;Check for 0 or 1
0A4C	DA 0993		2354	JC RIPPL1	;Jump if 0
0A4F	CA 09C0		2355	JZ MARCH3	;Jump if 1
0A52	FE 03		2356	CPI 3	;Check for 2, 3 or more
0A54	DA 09D1		2357	JC MARCH4	;Jump if 2
0A57	CA 09FA		2358	JZ MARCH7	;Jump if 3
0A5A	C3 0A0B		2359	JMP MARCH8	;Jump by default
			2360		
			2361		
0A5D			2362	ST_DONE:	
0A5D	21 0A61		2363	LXI H,1\$	;Point to the middle digit of version#
0A60	FF		2364	RST 7	;Go print it
0A61	31		2365	1\$: DB MIDDLE_NUMBER	;Middle digit of version number.
0A62	21 0A66		2366	LXI H,2\$	;Point to low digit of version#
0A65	FF		2367	RST 7	;Go print it
0A66	31		2368	2\$: DB LOW_NUMBER	;Low digit of version number.
			2369		;Now we fall through to the rest of the ROM code
0A67	C3 0024		2370	JMP INIT_VECTOR	;Get back into the Low ROM.
			2371		
			2372	;*****	
			2373	; RAM_LOAD	
			2374	; RAM_LOAD	
			2375	; RAM_LOAD	
0A6A			2376	RAM_LOAD:	
0A6A	CD 0104		2377	CALL SAVESP	;Go zero the success code and save the SP
0A6D	CD 0095		2378	CALL TU_SEND_PACKET	;Go send a packet.
0A70	21 4100		2379	LXI H,START	;Point to the destination area

Error	Addr	Code	Seq	Source statement	
	0A73	22 40A3	2380	SHLD LODADR	;Save the address
	0A76	3E 18	2381	MVI A,024	;This is the number of 128 byte packets
	0A78	32 40A8	2382	STA PAKCNT	;Save the count
	0A7B	D7	2383	RST 2	;Go get the first character
	0A7C	FE 01	2384	CPI 01	;Is the Flag byte good?
	0A7E	C4 0081	2385	CNZ TUERR3	;Leave if not
	0A81	32 409E	2386	STA WORD_SUM	;This must be included in the checksum
	0A84	D7	2387	RST 2	;Go get the byte count
	0A85	FE 80	2388	CPI 128	;Is it OK?
	0A87	C4 0081	2389	CNZ TUERR3	;Leave if not
	0A8A	32 409F	2390	STA WORD_SUM+1	;Include this in the checksum
	0A8D	32 40A7	2391	STA LODCNT	;Save the count
	0A90	CD 0AF2	2392	CALL TU_RCV_PACKETS	;Go get the packet
	0A93	21 40A8	2393	LXI H,PAKCNT	;Point to the Packet Count
	0A96	35	2394	DCR M	;Subtract one
	0A97	C2 0A7B	2395	JNZ 1\$	;Loop if not done
	0A9A	21 40AA	2396	LXI H,ENDPAK	;Point to the destination of the End Packet
	0A9D	22 40A3	2397	SHLD LODADR	;Save the address
	0AA0	D7	2398	RST 2	;Get the Flag byte of the End Packet
	0AA1	FE 02	2399	CPI 2	;Is it OK?
	0AA3	C4 0081	2400	CNZ TUERR3	;Error if not
	0AA6	32 409E	2401	STA WORD_SUM	;Need to include this in the checksum
	0AA9	D7	2402	RST 2	;Go get the byte count
	0AAA	FE 0A	2403	CPI 10	;Check the byte count
	0AAC	C4 0081	2404	CNZ TUERR3	;Leave if incorrect
	0AAF	32 409F	2405	STA WORD_SUM+1	;Include in the checksum
	0AB2	32 40A7	2406	STA LODCNT	;Save the count
	0AB5	CD 0AF2	2407	CALL TU_RCV_PACKETS	;Go get the End Packet
	0AB8	21 40AA	2408	LXI H,ENDPAK	;Point to the End Packet again
	0ABB	7E	2409	MOV A,M	;Get the first byte
	0ABC	FE 40	2410	CPI 040H	;Check the Op Code
	0ABE	C4 0081	2411	CNZ TUERR3	;Leave if incorrect
	0AC1	23	2412	INX H	;And bump the pointer again
	0AC2	7E	2413	MOV A,M	;and get the success code
	0AC3	17	2414	RAL	;Check for a negative number
	0AC4	DC 0081	2415	CC TUERR3	;Error if it is
	0AC7	3A 4091	2416	LDA UNIT	;Get the Unit number
	0ACA	23	2417	INX H	;Bump the pointer
	0ACB	BE	2418	CMP M	;Check for the correct unit number
	0ACC	C4 0081	2419	CNZ TUERR3	;Error if not the same
	0ACF	23	2420	INX H	;Point to the next byte
	0AD0	3A 4092	2421	LDA SNDPAK+5	;Get the Switch Byte
	0AD3	BE	2422	CMP M	;Is it the same?
	0AD4	C4 0081	2423	CNZ TUERR3	;Error if not
	0AD7	23	2424	INX H	;Point to the next byte
	0ADB	7E	2425	MOV A,M	;Get it
	0AD9	23	2426	INX H	;Point to the next byte
	0ADA	B6	2427	ORA M	;OR it in
	0ADB	23	2428	INX H	;One more to go
	0ADC	B6	2429	ORA M	;Now let's check

Error	Addr	Code	Seq	Source statement
	0ADD	C4 0081	2430	CNZ TUERR3 ;It is an error if the result isn't 0
	0AE0	23	2431	INX H ;Point to the high byte of the byte count
	0AE1	7E	2432	MOV A,M ;and put it in the Accumulator
	0AE2	FE 0C	2433	CPI 0CH ;Check it
	0AE4	C4 0081	2434	CNZ TUERR3 ;Leave if it isn't right
	0AE7	23	2435	INX H ;Point to the next byte
	0AE8	7E	2436	MOV A,M ;Bring it in
	0AE9	23	2437	INX H ;and the last byte
	0AEA	B6	2438	ORA M ;and OR it in
	0AEB	C4 0081	2439	CNZ TUERR3 ;Leave if the last two bytes aren't 0
	0AEE	32 4116	2440	STA SUCCES ;Use the zero in A to clear out the Success Code
			2441	;byte in case the boot block was garbage and we
			2442	;messed up SUCCES
	0AF1	C9	2443	RET ;Leave
			2444	;
			2445	;*****
			2446	;
			2447	;*****
			2448	;
			2449	; TU_RCV_PACKETS
			2450	;
	0AF2		2451	TU_RCV_PACKETS:
	0AF2	D7	2452	RST 2 ;Go look for a character
	0AF3	21 40A7	2453	LXI H,LODCNT ;Point to the count
	0AF6	CD 00DE	2454	CALL CHKSUM_WORD ;Go to the Word Checksum routine
	0AF9	2A 40A3	2455	LHLD LODADR ;Get the address for the data
	0AFC	7C	2456	MOV A,H ;Get the high byte of the address
	0AFD	FE 4B	2457	CPI 4BH ;Looking for 4B80H (beginning of silo)
	0AFF	C2 0B08	2458	JNZ 1\$;Jump over next part if not
	0B02	7D	2459	MOV A,L ;Get the low byte of the address
	0B03	FE 80	2460	CPI 80H ;Check it
	0B05	CA 0B0D	2461	JZ 2\$;If we are at the silo, skip the write
	0B08	70	2462	1\$: MOV M,B ;Send the data out
	0B09	23	2463	INX H ;Bump the pointer
	0B0A	22 40A3	2464	SHLD LODADR ;Save the address
	0B0D	21 40A7	2465	2\$: LXI H,LODCNT ;Point to the byte count
	0B10	35	2466	DCR M ;Decrement the count
	0B11	C2 0AF2	2467	JNZ TU_RCV_PACKETS ;Loop if not done
	0B14	D7	2468	RST 2 ;Go get the low byte of the checksum
	0B15	21 409E	2469	LXI H,WORD_SUM ;Point to the low byte of the checksum
	0B18	BE	2470	CMP M ;Check it
	0B19	C4 0081	2471	CNZ TUERR3 ;Error if not equal
	0B1C	D7	2472	RST 2 ;Get the high byte
	0B1D	21 409F	2473	LXI H,WORD_SUM+1 ;Point to the high byte of the checksum
	0B20	BE	2474	CMP M ;Is it equal to the calculated
	0B21	C8	2475	RZ ;Leave if OK
	0B22	CD 0081	2476	CALL TUERR3 ;Error if not
			2477	;
			2478	;*****
			2479	;

Error	Addr	Code	Seq	Source statement
			2480	
			2481	
			2482	
			2483	*****
			2484	:
			2485	:
0B25			2486	SEND_TRANS_TEXT:
0B25	3A	7E49	2487	LDA CLMSG_COUNT ;Get the Connection Lost Message count.
0B28	A7		2488	ANA A ;Check it
0B29	C2	0B48	2489	JNZ SEND_CON_LOST ;IF NOT 0, THEN jump.
0B2C	2A	7E44	2490	LHLD MSG15_PTR ;ELSE, we must be here because of a MSG
			2491	;Type 15, so get the pointer.
0B2F	CD	0B57	2492	CALL SPECIAL_SEND ;Try to send it.
0B32	C0		2493	RNZ ;Leave if we didn't send it.
0B33	22	7E44	2494	SHLD MSG15_PTR ;Save the new pointer.
0B36	21	7E46	2495	LXI H,MSG15_COUNT ;Point to the count
0B39	35		2496	DCR M ;and decrement it.
0B3A	C0		2497	RNZ ;IF NOT 0, THEN leave (not done yet).
0B3B	3A	7E04	2498	LDA TALK_FLAG ;Get the Talk Flag.
0B3E	1F		2499	RAR ;Are we in Talk Mode?
0B3F	D8		2500	RC ;IF TALK, THEN leave.
0B40	3A	7E10	2501	LDA RDCON ;ELSE, do we have a connection?
0B43	A7		2502	ANA A ;Well?
0B44	C2	1016	2503	JNZ TSQA_VECTOR ;IF RDCON = -TALK, THEN see about sending ^Q.
0B47	C9		2504	RET ;ELSE, leave.
			2505	
0B48			2506	SEND_CON_LOST:
0B48	2A	7E47	2507	LHLD CLMSG_PTR ;Get the pointer into the Connection Lost Msg.
0B4B	CD	0B57	2508	CALL SPECIAL_SEND ;Go try to send it.
0B4E	C0		2509	RNZ ;Leave if we couldn't send it.
0B4F	22	7E47	2510	SHLD CLMSG_PTR ;Store the updated pointer.
0B52	21	7E49	2511	LXI H,CLMSG_COUNT ;Point to the Connection Lost Count.
0B55	35		2512	DCR M ;Subtract 1
0B56	C9		2513	RET ;Leave.
			2514	
0B57			2515	SPECIAL_SEND:
0B57	3A	40CD	2516	LDA S_FLAG ;Get the Control S flag
0B5A	A7		2517	ANA A ;Is it set?
0B5B	C0		2518	RNZ ;If set, leave.
0B5C	3A	40BB	2519	LDA O_FLAG ;Get the Control O flag
0B5F	A7		2520	ANA A ;Is it set?
0B60	C2	0B6D	2521	JNZ 1\$;Jump if set
0B63	DB	49	2522	IN TTSR ;Get the Console Terminal SR
0B65	E6	01	2523	ANI 1 ;Save only the Xmit Rdy bit.
0B67	EE	01	2524	XRI 1 ;Check for it this way so NOT XMIT RDY will
			2525	;generate a non zero result, and XMIT RDY will
			2526	;generate a zero result.
0B69	C0		2527	RNZ ;Now if we leave it will indicate that the
			2528	;character was not sent.
0B6A	7E		2529	MOV A,M ;Get the data

Error	Addr	Code	Seq	Source statement	
	0B6B	D3 48	2530	OUT TTDB	;Send it out
	0B6D	23	2531	1\$: INX H	;Bump the pointer ahead.
	0B6E	AF	2532	XRA A	;Set the Z bit to indicate that we sent the
	0B6F	C9	2533	RET	;character and leave.
			2534		
			2535		
			2536		
			2537	;*****	
			2538	;	
	0B70		2539	RD_LD_COMMAND:	
	0B70	3A 408B	2540	LDA CODE_FLAG	;Get the Code flag (0=Console, 1=Micmon).
	0B73	21 40D3	2541	LXI H,RUN_FLAG	;Point to the Run Flag (indicates PM).
	0B76	B6	2542	ORA M	;OR it in.
	0B77	EE 01	2543	XRI 1	;Flip it (both must have been 0 to generate
			2544		;a one at this point).
	0B79	21 4080	2545	LXI H,RAM_VALID	;Point to the 8085 RAM Valid flag.
	0B7C	A6	2546	ANA M	;AND it in.
	0B7D	21 40D5	2547	LXI H,WCS_LOAD_FLAGS	;Point to the MicroCode Valid Flag.
	0B80	A6	2548	ANA M	;AND them together.
	0B81	47	2549	MOV B,A	;Store the result in B for a moment.
	0B82	DB 03	2550	IN SECURE	;Get the Secure position of the switch.
	0B84	A0	2551	ANA B	;AND it with the rest.
	0B85	1F	2552	RAR	;What is the result?
	0B86	D2 0BD9	2553	JNC RD_LOAD_ERROR	;IF NOT(CONSOLE * WCS_LOAD_FLAGS * -(Program
			2554		;Mode) * -Secure), THEN leave.
	0B89	3A 7E40	2555	LDA IBUF_CNT_COPY	;ELSE, get the count.
	0B8C	D6 06	2556	SUI 6	;Reduce it by the 1 MSG TYPE byte + 5 Address
			2557		;Field bytes.
	0B8E	CA 1001	2558	JZ UB_VECTOR	;IF COUNT = 0, THEN leave.
	0B91	FA 1001	2559	JM UB_VECTOR	;IF COUNT < 0, THEN leave.
	0B94	32 7E4F	2560	STA RD_LD_DCOUNT	;Save the data count here.
	0B97	AF	2561	XRA A	;Make a 0
	0B98	32 7E53	2562	STA RD_LD_OE	;Clear the Odd/Even flag.
	0B9B	3C	2563	INR A	;Make a 1.
	0B9C	32 7E2C	2564	STA RD_LD_FLAG	;Set this flag.
	0B9F	32 7E1F	2565	STA MSG_TO_READ	;Set the Message to Read flag.
	0BA2	3E 0A	2566	MVI A,10	;Set up a count of 10
	0BA4	32 7E4C	2567	STA RD_LD_CCOUNT	;Prime this count
	0BA7	32 7E52	2568	STA RD_LD_ECOUNT	;and this one also.
	0BAA	47	2569	MOV B,A	;Put the 10 in B.
	0BAB	11 0D7E	2570	LXI D,RD_LD_CSTRING	;Point to the Command String as the source.
	0BAE	21 7E55	2571	LXI H,RD_LD_CBUFFER	;and the Command Buffer as the destination.
	0BB1	22 7E4A	2572	SHLD RD_LD_CPTR	;Save this address here.
	0BB4	CD 00FB	2573	CALL COPY1	;Go copy them over.
	0BB7	2A 7E42	2574	LHLD IBUF_ADR_COPY	;Get this pointer.
	0BBA	EB	2575	XCHG	;Put it in DE.
	0BBB	13	2576	INX D	;Bump it ahead by two.
	0BBC	13	2577	INX D	;
	0BBD	21 7E5A	2578	LXI H,RD_LD_CBUFFER+5	;Point to this destination.
	0BC0	06 04	2579	MVI B,4	;Set up a count of 4.

Error	Addr	Code	Seq	Source statement
OBC2	1A		2580	1\$: LDAX D ;Get the source.
OBC3	77		2581	MOV M,A ;Store it.
OBC4	13		2582	INX D ;Bump the source pointer.
OBC5	2B		2583	DCX H ;and the Destination pointer.
OBC6	05		2584	DCR B ;Decrement the count.
OBC7	C2 OBC2		2585	JNZ 1\$;IF NOT 0, THEN loop back.
OBCA	EB		2586	XCHG ;Put the address where the data starts into HL.
OBCB	22 7E4D		2587	SHLD RD_LD_DPTR ;Save the address here.
OBCE	21 7E4F		2588	LXI H,RD_LD_DCOUNT ;Point to the Data Count.
OBD1	7E		2589	MOV A,M ;Get it.
OBD2	32 7E5B		2590	STA RD_LD_CBUFFER+6 ;Store it here.
OBD5	1F		2591	RAR ;Is it odd or even?
OBD6	D0		2592	RNC ;IF EVEN, THEN leave.
OBD7	34		2593	INR M ;ELSE, make it even
OBD8	C9		2594	RET ;and leave.
			2595	
			2596	
			2597	
OBD9			2598	RD_LOAD_ERROR:
OBD9	21 7E2D		2599	LXI H,RD_LD_ERR ;Point to the RD Load Error flag.
OBDC	36 80		2600	MVI M,80H ;Set it.
OBDE	C3 1001		2601	JMP UB_VECTOR ;Go clean things up.
			2602	;
			2603	;.....
			2604	
			2605	
			2606	
			2607	;.....
			2608	;
			2609	; DCD_MESSAGE Message Type 14
			2610	;
			2611	; The first byte after the message type is the Switches byte. Bits 6 and
			2612	; 7 indicate how the bits are to be used, and bits 0-5 indicate the bit
			2613	; switches to be affected.
			2614	;
			2615	; BITS: 7 6
			2616	;
			2617	; 0 0 Set to value
			2618	; 0 1 Set specified bits
			2619	; 1 0 Clear specified bits
			2620	; 1 1 No-op; ignore byte
			2621	;
			2622	;
			2623	; BITS: 0 Enable Local Copy mode
			2624	; 1 Enable Local Control mode
			2625	; 2 Enable Talk Echo
			2626	; 3 Not used
			2627	; 4 Enable Talk
			2628	; 5 Discard Remote Output (while connection lost)
			2629	;

Error	Addr	Code	Seq	Source statement
			2630	; The second byte contains the Override Local Copy in Secure bit.
			2631	;
			2632	;
			2633	;
OBE1			2634	DCD_MESSAGE:
OBE1	21	7E40	2635	LXI H,IBUF_CNT_COPY ;Point at the Input Buffer count
OBE4	35		2636	DCR M ;Subtract 1.
OBE5	CA	1001	2637	JZ UB_VECTOR ;IF COUNT = 0, THEN leave.
OBE8	3A	7E04	2638	LDA TALK_FLAG ;Get the Talk Flags
OBE8	32	7E73	2639	STA TALK_COPY ;Save a copy for later in DCD_FINISH.
OBE8	2A	7E42	2640	LHLD IBUF_ADR_COPY ;Get the pointer into the buffer
OBF1	23		2641	INX H ;Bump it ahead.
OBF2	46		2642	MOV B,M ;Get the Control Data byte.
OBF3	22	7E42	2643	SHLD IBUF_ADR_COPY ;Save the updated address.
OBF6	16	00	2644	MVI D,0 ;Set up D with 0.
OBF8	78		2645	MOV A,B ;Put a copy of first DCD byte in A to work on.
OBF9	E6	C0	2646	ANI 0C0H ;Save bits 6 and 7.
OBF8	CA	0CB1	2647	JZ DCD_STV ;If zero, go to the Set to Value flow.
OBF8	FE	80	2648	CPI 80H ;How about 01 or 10?
OC00	DA	0C94	2649	JC DCD_SSB ;IF 01, THEN go to the Set Specified Bits flow.
OC03	CA	0C31	2650	JZ DCD_CSB ;IF 10, THEN go to Clear Specified Bits.
			2651	;ELSE, it is a nop, so fall through.
OC06			2652	DCD_MESS0:
OC06	21	7E40	2653	LXI H,IBUF_CNT_COPY ;Point to the count.
OC09	35		2654	DCR M ;Check for another byte.
OC0A	CA	1001	2655	JZ UB_VECTOR ;IF NO MORE BYTES, THEN leave.
OC0D	2A	7E42	2656	LHLD IBUF_ADR_COPY ;Get the address.
OC10	23		2657	INX H ;Bump it ahead.
OC11	7E		2658	MOV A,M ;Get the byte
OC12	FE	02	2659	CPI 2 ;See if the byte STV(0) or STV(1)
OC14	DA	0C27	2660	JC 1\$;IF (A=0) OR (A=1), THEN jump.
OC17	16	00	2661	MVI D,0 ;Clear out D.
OC19	FE	81	2662	CPI 81H ;Is it CSB*Override Bit?
OC1B	CA	0C2A	2663	JZ 2\$;IF YES, THEN jump.
OC1E	14		2664	INR D ;ELSE, make a 1 in D.
OC1F	FE	41	2665	CPI 41H ;Is it SSB*Override Bit?
OC21	CA	0C2A	2666	JZ 2\$;IF YES, THEN jump.
OC24	C3	1001	2667	JMP UB_VECTOR ;ELSE, nop, so leave.
OC27	E6	01	2668	1\$: ANI 1 ;Save only the LSB.
OC29	57		2669	MOV D,A ;Put it in D and fall through.
OC2A	21	7E67	2670	2\$: LXI H,OVERWRITE_LCS ;SET/CLEAR (depends on contents of D).
OC2D	72		2671	MOV M,D ;Do it.
OC2E	C3	1001	2672	JMP UB_VECTOR ;Leave.
			2673	
OC31			2674	DCD_CSB:
OC31	CD	0CD9	2675	CALL DCD_SHARE ;Go share some code.
OC34	78		2676	MOV A,B ;Get the DCD byte.
OC35	E6	14	2677	ANI 14H ;See if the Talk or Talk Echo bits are set
OC37	CA	0C4C	2678	JZ DCD_CSB1 ;Jump if neither set
OC3A	1E	02	2679	MVI E,2 ;Set up a 2 in E.

Error	Addr	Code	Seq	Source statement	
	0C3C	FE 10	2680	CPI 10H	;Is it only Clear Talk?
	0C3E	CA 0C46	2681	JZ 3\$	;IF CLEAR TALK * -CLEAR ECHO, THEN leave E=2
	0C41	DA 0C45	2682	JC 2\$	;IF CLEAR ECHO * -CLEAR TALK, THEN make a 1.
	JC44	1D	2683	DCR E	;IF CLEAR TALK * CLEAR ECHO, THEN make a 0.
	0C45	1D	2684	2\$: DCR E	;Make a 1.
	0C46	7B	2685	3\$: MOV A,E	;Get the contents of E.
	0C47	21 7E04	2686	LXI H,TALK_FLAG	;Point to the Talk and Talk Echo flags
	0C4A	A6	2687	ANA M	;AND in the mask
	0C4B		2688	DCD_CSBO:	
	0C4B	77	2689	MOV M,A	;Send it back out
	0C4C		2690	DCD_CSBI:	
	0C4C	21 7E71	2691	LXI H,LOCAL_COPY	;Point to the Local Copy bit
	0C4F	78	2692	MOV A,B	;Get the DCD byte again
	0C50	1F	2693	RAR	;See if we have (Clear/Set) Local Copy.
	0C51	D2 0C55	2694	JNC 1\$	;IF NOT (CLEAR/SET) LOCAL COPY, THEN jump.
	0C54	72	2695	MOV M,D	;Clear or set the flag depending on what is in
			2696		;register D.
	0C55	1F	2697	1\$: RAR	;How about (Clear) Parallel Control.
	0C56	D2 0C5B	2698	JNC DCD_FINISH	;Go build the LR_MASK based on what is in the
			2699		;Local_Copy and Parallel_Control flags.
	0C59	23	2700	INX H	;Point to the Parallel Control flag.
	0C5A	72	2701	MOV M,D	;Clear or Set it.
	0C5B		2702	DCD_FINISH:	
	0C5B	3A 7E72	2703	LDA PARALLEL_CNTRL	;Get this flag.
	0C5E	A7	2704	ANA A	;Is it set?
	0C5F	3E A5	2705	MVI A,0ASH	;Assume it is.
	0C61	C2 0C6E	2706	JNZ 1\$	;IF SET, THEN jump.
	0C64	3A 7E71	2707	LDA LOCAL_COPY	;How about Local Copy?
	0C67	A7	2708	ANA A	;Is it set?
	0C68	3E 25	2709	MVI A,25H	;Assume it is.
	0C6A	C2 0C6E	2710	JNZ 1\$	;IF SET, THEN jump.
	0C6D	3D	2711	DCR A	;ELSE, make a 24H in A.
	0C6E	32 40BE	2712	1\$: STA LR_MASK	;This is the new LR Mask.
	0C71	3A 7E73	2713	LDA TALK_COPY	;Now get the Talk Flag copy.
	0C74	E6 01	2714	ANI 1	;Save only the Talk bit here.
	0C76	47	2715	MOV B,A	;Put it here.
	0C77	3A 7E04	2716	LDA TALK_FLAG	;Get the Talk Flags
	0C7A	E6 01	2717	ANI 1	;Save only the Talk bit.
	0C7C	B8	2718	CMP B	;Are they the same?
	0C7D	CA 0C06	2719	JZ DCD_MESS0	;IF EQUAL, THEN leave.
	0C80	A7	2720	ANA A	;Is it set or clear?
	0C81	C2 0C8E	2721	JNZ 2\$	;Jump if entering Talk Mode.
	0C84	3A 7E46	2722	LDA MSG15_COUNT	;See if we are doing a Message Type 15.
	0C87	A7	2723	ANA A	;If the count is 0, we aren't.
	0C88	CC 1016	2724	CZ TSQA_VECTOR	;IF NOT MSG TYPE 15, THEN see if we should send
			2725		;an ^Q.
	0C8B	C3 0C06	2726	JMP DCD_MESS0	;Go see if there is another byte of DCD stuff.
	0C8E	CD 1010	2727	2\$: CALL TSS_VECTOR	;Go see if we should send ^S.
	0C91	C3 0C06	2728	JMP DCD_MESS0	;Check for another byte.
			2729		

Error	Addr	Code	Seq	Source statement
	0C94		2730	DCD_SSB:
	0C94	14	2731	INR D ;Make a 1 in D
	0C95	5A	2732	MOV E,D ;Make a 1 in E also.
	0C96	CD 0CD9	2733	CALL DCD_SHARE ;Go see about setting the Discard Remote flag.
	0C99	78	2734	MOV A,B ;Get the DCD byte back
	0C9A	E6 14	2735	ANI 14H ;Check for Talk Echo or Talk
	0C9C	CA 0C4C	2736	JZ DCD_CSB1 ;If neither set, don't change anything.
	0C9F	FE 10	2737	CPI 10H ;Is it Talk?
	0CA1	CA 0CA9	2738	JZ 2\$;If it is, jump.
	0CA4	DA 0CA8	2739	JC 1\$;IF SET ECHO, THEN jump to make a 2.
	0CA7	1C	2740	INR E ;ELSE TALK * ECHO, so make a 3.
	0CA8	1C	2741	1\$: INR E ;
	0CA9	7B	2742	2\$: MOV A,E ;Get the Constant from E.
	0CAA	21 7E04	2743	LXI H,TALK_FLAG ;Point to the Talk Flag byte
	0CAD	B6	2744	ORA M ;OR in the old with the new.
	0CAE	C3 0C4B	2745	JMP DCD_CSB0 ;Go share some code.
			2746	
			2747	
	0CB1		2748	DCD_STV:
	0CB1	78	2749	MOV A,B ;Get back an unmodified copy
	0CB2	E6 20	2750	ANI 20H ;Find out if bit 5 is set or clear
	0CB4	07	2751	RLC ;Do this three times to get the bit into bit 0
	0CB5	07	2752	RLC ;
	0CB6	07	2753	RLC ;
	0CB7	32 7E05	2754	STA DISCARD_RO ;Set or clear the Discard Remote bit
	0CBA	78	2755	MOV A,B ;Get the DCD byte back
	0CBB	E6 14	2756	ANI 14H ;Save only the Talk and Talk Echo flags.
	0CBD	1F	2757	RAR ;Put the Talk Echo flag into bit 1.
	0CBE	4F	2758	MOV C,A ;Store this here for a moment.
	0CBF	1F	2759	RAR ;We need three more rotates to get the Talk flag
	0CC0	1F	2760	RAR ;into bit 0.
	0CC1	1F	2761	RAR ;This is the third.
	0CC2	B1	2762	ORA C ;Build Talk in bit 0, Talk Echo in bit 1.
	0CC3	E6 03	2763	ANI 3 ;Save only the Talk and Talk Echo bits.
	0CC5	21 7E04	2764	1\$: LXI H,TALK_FLAG ;Point to the Talk Flag byte
	0CC8	77	2765	MOV M,A ;Send out the correct stuff
	0CC9	78	2766	MOV A,B ;Get the DCD byte again
	0CCA	E6 01	2767	ANI 1 ;Save only the Local Copy bit
	0CCC	32 7E71	2768	STA LOCAL_COPY ;and store it.
	0CCF	78	2769	MOV A,B ;Get the DCD one last time.
	0CD0	E6 02	2770	ANI 2 ;Save bit 1, the Parallel Control bit.
	0CD2	1F	2771	RAR ;Put it in bit 00
	0CD3	32 7E72	2772	STA PARALLEL_CNTRL ;and store it.
	0CD6	C3 0C5B	2773	JMP DCD_FINISH ;Go clean things up and leave.
			2774	
	0CD9		2775	DCD_SHARE:
	0CD9	78	2776	MOV A,B ;Get the DCD byte
	0CDA	E6 20	2777	ANI 20H ;Check the Discard Remote bit
	0CDC	C8	2778	RZ ;IF NOT SET, THEN leave.
	0CDD	21 7E05	2779	LXI H,DISCARD_RO ;Point to the Discard Remote Flag byte

Error	Addr	Code	Seq	Source statement
	OCE0	72	2780	MOV M,D ;Zero it
	OCE1	C9	2781	RET ;Leave.
			2782	:
			2783	:.....
			2784	:
			2785	:.....
			2786	:
			2787	:
			2788	: TEXT_MESSAGE Message Type 15
			2789	:
OCE2			2790	TEXT_MESSAGE:
OCE2	3A 7E40		2791	LDA IBUF_CNT_COPY ;Get the Input Buffer Count Copy
OCE5	3D		2792	DCR A ;Decrement the count.
OCE6	CA 1001		2793	JZ UB_VECTOR ;IF COUNT=0, THEN cleanup and leave.
OCE9	32 7E46		2794	STA MSG15_COUNT ;ELSE, there is something to print.
OCEC	2A 7E42		2795	LHLD IBUF_ADR_COPY ;Get the address.
OCEF	23		2796	INX H ;Bump it ahead.
OCF0	22 7E44		2797	SHLD MSG15_PTR ;Store it here.
OCF3	3E 01		2798	MVI A,1 ;Make a 1.
OCF5	32 7E1F		2799	STA MSG_TO_READ ;Set this flag.
OCF8	C3 111D		2800	JMP TESTSET_S ;Go see about send ^S.
			2801	:
			2802	:.....
			2803	:
			2804	:.....
			2805	:
			2806	:
			2807	: TALK
			2808	:
OCFB	3A 7E13		2809	TALK: LDA TALKSR ;Get the Talk SR
OCFE	1F		2810	RAR ;Is the Ready bit set?
OCFF	D2 0D0F		2811	JNC TALK_SEND ;Jump if not (we have a character to be sent)
OD02	CD 0046		2812	CALL GS_VECTOR ;Go look for a character
OD05	D0		2813	RNC ;Leave if none found
OD06	21 7E14		2814	LXI H,TALKDB ;Point to this DB
OD09	70		2815	MOV M,B ;Store the character there
OD0A	2B		2816	DCX H ;Point to the Talk SR
OD0B	79		2817	MOV A,C ;Get the source flag
OD0C	E6 C0		2818	ANI 0C0H ;Clear the Xmit Rdy bit
OD0E	77		2819	MOV M,A ;Store this SR
OD0F			2820	TALK_SEND:
OD0F	3A 7E04		2821	LDA TALK_FLAG ;Get the Talk Flags
OD12	E6 02		2822	ANI 2 ;See if Talk Echo or Talk No Echo
OD14	CA 0D2D		2823	JZ TALK_NO_ECHO ;Jump if Talk No Echo
OD17	DB 49		2824	IN TTSR ;Get the Local SR
OD19	21 7E02		2825	LXI H,REMXSR ;Point to the Pseudo Remote SR
OD1C	A6		2826	ANA M ;AND them together
OD1D	C8		2827	RZ ;Leave if both not set
OD1E	36 00		2828	MVI M,0 ;Zero the Xmit Rdy in the Remote SR
OD20	23		2829	INX H ;Point to the Remote DB

Error	Addr	Code	Seq	Source statement	
0D21	EB		2830	XCHG	;Put the address in DE
0D22	21 7E14		2831	LXI H,TALKDB	;Point to the Talk DB
0D25	7E		2832	MOV A,M	;Get the character
0D26	D3 48		2833	OUT TTDB	;Send it to Local
0D28			2834	TALK_FIN0:	
0D28	12		2835	STAX D	;Store the character in the Remote DB
0D29			2836	TALK_FIN1:	
0D29	2B		2837	DCX H	;Point to the Talk SR
0D2A	36 01		2838	MVI M,1	;Set the Xmit Rdy
0D2C	C9		2839	RET	;Leave
			2840		
0D2D			2841	TALK_NO_ECHO:	
0D2D	21 7E13		2842	LXI H,TALKSR	;Point to the Talk SR
0D30	7E		2843	MOV A,M	;Get it
0D31	23		2844	INX H	;Now point to the Talk DB
0D32	17		2845	RAL	;Where did the character come from?
0D33	DA 0D43		2846	JC 1\$	;Jump if it was from Remote
0D36	11 7E02		2847	LXI D,REMXSR	;See if the Remote port is ready
0D39	1A		2848	LDAX D	;Get the SR
0D3A	1F		2849	RAR	;Is it ready?
0D3B	D0		2850	RNC	;Leave if not
0D3C	AF		2851	XRA A	;Make a 0
0D3D	12		2852	STAX D	;Zero the Xmit Rdy
0D3E	13		2853	INX D	;Point to the Remote DB
0D3F	7E		2854	MOV A,M	;Get the character
0D40	C3 0D28		2855	JMP TALK_FIN0	;Go share the clean up
0D43	DB 49		2856	1\$: IN TTSR	;Get the Local SR
0D45	1F		2857	RAR	;Is it Ready?
0D46	D0		2858	RNC	;Leave if not
0D47	7E		2859	MOV A,M	;Get the character
0D48	D3 48		2860	OUT TTDB	;Send it out
0D4A	C3 0D29		2861	JMP TALK_FIN1	;Go share the cleanup
			2862	:	
			2863	:	
			2864		
			2865		
			2866		
			2867	:	
			2868	:	
			2869	;BYTE 0:	
			2870	;BITS 07 06 05 04 03 02 01 00	
			2871	: NU NU Discard Talk NU Talk Local Local	
			2872	: Remote Output Echo Control Copy	
			2873	:	
			2874	:	
			2875	;BYTE 1:	
			2876	;BITS 07 06 05 04 03 02 01 00	
			2877	: NU NU NU NU NU NU Remote Remote	
			2878	: Secure	
			2879	:	

Error	Addr	Code	Seq	Source statement
			2880	;BYTE 2:
			2881	;BITS 07 06 05 04 03 02 01 00
			2882	: NU NU NU NU NU NU NU 00
			2883	: 1=0 sec Timeout
			2884	: 1=0 Sec Timeout
			2885	CTL_BIT_BUILD:
0D4D			2886	LDA PARALLEL_CNTRL ;Get this flag.
0D4D	3A 7E72		2887	RAL ;Shift it into bit position 1.
0D50	17		2888	LXI H,LOCAL_COPY ;Point to this flag.
0D51	21 7E71		2889	ORA M ;OR it in.
0D54	B6		2890	MOV C,A ;Store the result
0D55	4F		2891	LDA TALK_FLAG ;Get the Talk and Talk Echo flag byte
0D56	3A 7E04		2892	MOV B,A ;Put a copy in B
0D59	47		2893	ANI 1 ;Save the Talk Flag
0D5A	E6 01		2894	JZ 1\$;Jump if not in Talk Mode
0D5C	CA 0D63		2895	MVI A,10H ;Set bit 4
0D5F	3E 10		2896	ORA C ;OR it into the first CTL byte
0D61	B1		2897	MOV C,A ;Put the result back
0D62	4F		2898	MOV A,B ;Get the Talk byte
0D63	78		2899	ANI 2 ;Save the Talk Echo bit
0D64	E6 02		2900	RLC ;Rotate it into bit 2
0D66	07		2901	ORA C ;OR it into CTL byte 1
0D67	B1		2902	MOV C,A ;Put the result back
0D68	4F		2903	LDA DISCARD_RO ;Get the Discard Remote bit
0D69	3A 7E05		2904	ANA A ;Is it set?
0D6C	A7		2905	MOV A,C ;Get the accumulated CTL byte 1
0D6D	79		2906	JZ 2\$;Jump if not
0D6E	CA 0D73		2907	ORI 20H ;Set bit 5
0D71	F6 20		2908	STA CTL_BUFFER+1 ;Store the second CTL byte
0D73	32 7E65		2909	IN SECURE ;Look at the Secure bit of the Keyswitch.
0D76	DB 03		2910	ANI 1 ;Mask off the rest
0D78	E6 01		2911	STA CTL_BUFFER+2 ;Store it for transmission
0D7A	32 7E66		2912	RET ;
0D7D	C9		2913	:
			2914	:
			2915	END OF CTL BIT BUILD ROUTINE
			2916	:
			2917	
			2918	:
			2919	:
0D7E			2920	RD_LD_CSTRING:
0D7E	04 00		2921	DB 4,0 ;
0D80	0000 0000		2922	DW 0,0,0,0 ;
0D84	0000 0000		2923	:
			2924	:
			2925	:
			2926	:
			2927	
			2928	:

Error	Addr	Code	Seq	Source statement
			2929	; Version number codes.
			2930	;
		=0043	2931	C_LETTER EQU 'C' ;
		=004F	2932	O_LETTER EQU 'O' ;
		=004E	2933	N_LETTER EQU 'N' ;
		=0056	2934	V_LETTER EQU 'V' ;
		=0030	2935	HIGH_NUMBER EQU '0' ;High digit of version number
		=0031	2936	MIDDLE_NUMBER EQU '1' ;Middle digit of version number
		=0031	2937	LOW_NUMBER EQU '1' ;Low digit of version number
			2938	;
			2939	;.....
			2940	
			2941	
			2942	;.....
			2943	;
			2944	; MSG_DECODE
			2945	;
			2946	; This is the Message Decode routine. The first thing we do is call the
			2947	Decode Subroutine (DECSUB). By doing it this way the code that handles
			2948	each message type can do a RETURN at the end and get back here.
			2949	;
0D88			2950	MSG_DECODE:
0D88	CD 100A		2951	CALL RDD_VECTOR ;Go to the decode subroutine
			2952	;
			2953	; We return here after we finish decoding and executing the command.
			2954	; Next we prime an SOH header and the BLDBLK. Whether we are going to
			2955	send a Message Type 13 or 17 doesn't matter for most of this, so we
			2956	build a type 17, then check the RDSPER flag. If it is set, Type 17 is
			2957	correct and we jump over the code that would change it. If RDSPER is
			2958	clear, we fall through and execute the code that:
			2959	;
			2960	;
			2961	1) Checks for data in the output buffers.
			2962	2) Locks the correct buffer if there is something to be sent
			2963	3) Sets up the correct count in both the Header and BLDBLK.
			2964	4) Checks the CTL flag and builds the CTL bytes if necessary.
			2965	;
0D8B	21 7EA1		2965	LXI H,SOHHED ;Point to the header
0D8E	36 81		2966	MVI M,81H ;Set up an SOH
0D90	22 7E9A		2967	SHLD BLDBLK ;Save this address
0D93	23		2968	INX H ;Point to the second byte of the header
0D94	36 02		2969	MVI M,2 ;Set up a count of 1
0D96	23		2970	INX H ;Bump the address
0D97	36 00		2971	MVI M,0 ;Zero the high byte of the count in the header
0D99	23		2972	INX H ;Bump the pointer
0D9A	3A 7E2A		2973	LDA LAST_GOOD_MSG ;Get the Last Good Message number
0D9D	77		2974	MOV M,A ;Put it in the header
0D9E	3C		2975	INR A ;Bump the message number
0D9F	23		2976	INX H ;Bump the pointer
0DA0	77		2977	MOV M,A ;Store the Current Message number
0DA1	23		2978	INX H ;Point to the Station Number

Error	Addr	Code	Seq	Source statement	
	0DA2	3E 01	2979	MVI A,1	;Make a 1
	0DA4	77	2980	MOV M,A	;Make the Station Number 1
	0DA5	32 7E9D	2981	STA BLDBLK+3	;Set up a count of 1
	0DA8	21 17FE	2982	LXI H,MSG17_MESSAGE	;Point to the Message Type 17 Reply Message.
	0DAB	22 7E9E	2983	SHLD BLDBLK+4	;Store the address here.
	0DAE	3E 11	2984	MVI A,17	;Set up Message Type 17
	0DB0	32 7EA0	2985	STA BLDBLK+6	;Put it here.
	0DB3	3A 7E3B	2986	LDA RDSPER	;Get the RD Service Protocol Error flag
	0DB6	A7	2987	ANA A	;See if it is set
	0DB7	C2 0E0C	2988	JNZ 4\$	;IF SET, THEN jump.
	0DBA	3A 7E8E	2989	LDA OUTLOC	;ELSE, get the Locked Buffer Flag
	0DBD	A7	2990	ANA A	;Is there one?
	0DBE	C2 0DC9	2991	JNZ 1\$	;Jump if there is
	0DC1	2A 7E21	2992	LHLD ACTCNT	;Get the Active Buffer Count address
	0DC4	7E	2993	MOV A,M	;Get the count
	0DC5	A7	2994	ANA A	;Is it 0?
	0DC6	C4 100D	2995	CNZ FB0_VECTOR	;If there is something in the active buffer, go
			2996		;lock it.
	0DC9	11 7EA2	2997	LXI D,SOHHED+1	;Point to the low byte of the count
	0DCC	3A 7E8E	2998	LDA OUTLOC	;Get the Locked Buffer Flag
	0DCF	A7	2999	ANA A	;Is there a Locked Buffer?
	0DD0	CA 0DDA	3000	JZ 2\$	;Jump if not: AC = 0.
	0DD3	32 7E8F	3001	STA LBSIP	;Set the Locked Buffer Send In Progress flag
	0DD6	2A 7E25	3002	LHLD LOCCNT	;Get the address of the count
	0DD9	7E	3003	MOV A,M	;Get the count.
	0DDA	32 7E9D	3004	STA BLDBLK+3	;Store it in BLDBLK+3. If we came here from
			3005		;three lines up, the count will be 0.
	0DDD	47	3006	MOV B,A	;Put the count in B.
	0DDE	04	3007	INR B	;Make it one bigger.
	0DDF	3A 7E97	3008	LDA CTLFLG	;Get the Control Flag
	0DE2	32 7EA0	3009	STA BLDBLK+6	;Store it here now
	0DES	A7	3010	ANA A	;Check it
	0DE6	78	3011	MOV A,B	;Put the count in A
	0DE7	CA 0DEC	3012	JZ 3\$	;Jump if not set
	0DEA	C6 04	3013	ADI 4	;Add three if the CTL Flag is set
	0DEC	12	3014	STAX D	;
	0DED	13	3015	INX D	;Point to the high byte of the count
	0DEE	AF	3016	XRA A	;Make a 0
	0DEF	12	3017	STAX D	;and zero the high byte of the count
	0DF0	23	3018	INX H	;Build the address of the Buffer that goes with
			3019		;this count
	0DF1	22 7E9E	3020	SHLD BLDBLK+4	;Store it in the parameter block
	0DF4	3E 0D	3021	MVI A,13	;Make a MSG TYPE 13
	0DF6	21 7EA0	3022	LXI H,BLDBLK+6	;Point to this byte
	0DF9	B6	3023	ORA M	;OR the message type in
	0DFA	47	3024	MOV B,A	;Store it temporarily.
	0DFB	3A 7E70	3025	LDA CTL_C_INTR	;Get this flag.
	0DFE	4F	3026	MOV C,A	;Put it in C.
	0DFF	3A 7E2D	3027	LDA RD_LD_ERR	;Get this error flag.
	0E02	B0	3028	ORA B	;OR things together.

Error	Addr	Code	Seq	Source statement
0E03	B1		3029	ORA C
0E04	77		3030	MOV M,A
0E05	AF		3031	XRA A
0E06	32 7E70		3032	STA CTL_C_INTR
0E09	32 7E2D		3033	STA RD_LD_ERR
0E0C	AF		3034	4\$: XRA A
0E0D	CD 0E2E		3035	CALL BUILD
0E10	AF		3036	XRA A
0E11	32 7E1D		3037	STA MESSAG
0E14	32 7E1A		3038	STA DPFLAG
0E17	32 7E30		3039	STA RCRC
0E1A	32 7E31		3040	STA RCRC+1
0E1D	3C		3041	INR A
0E1E	32 7E98		3042	STA SOHENQ
0E21	21 7E68		3043	LXI H,HBUFFER
0E24	22 7E18		3044	SHLD IADDR
0E27	21 0008		3045	LXI H,8
0E2A	22 7E16		3046	SHLD ICNT
0E2D	C9		3047	RET
			3048	
			3049	
			3050	*****
			3051	*****
			3052	;
0E2E	CD 0E67		3053	BUILD: CALL BUILD_PREP
0E31	2A 7E9A		3054	LHLD BLDBLK
0E34	22 7E95		3055	SHLD HDADR
0E37	3A 7EA0		3056	LDA BLDBLK+6
0E3A	E6 20		3057	ANI 20H
0E3C	CA 0E4D		3058	JZ 1\$
0E3F	21 7E63		3059	LXI H,CTL_COUNT
0E42	36 04		3060	MVI M,4
0E44	23		3061	INX H
0E45	36 03		3062	MVI M,3
0E47	22 7E61		3063	SHLD CTL_PTR
0E4A	CD 0D4D		3064	CALL CTL_BIT_BUILD
0E4D	3A 7EA0		3065	1\$: LDA BLDBLK+6
0E50	32 7E77		3066	STA MSGTYP
0E53	3E 01		3067	MVI A,1
0E55	32 7E76		3068	STA MSGFLG
0E58	3A 7E9D		3069	LDA BLDBLK+3
0E5B	A7		3070	ANA A
0E5C	C8		3071	RZ
0E5D	32 7E27		3072	STA OUTCNT
0E60	2A 7E9E		3073	LHLD BLDBLK+4
0E63	22 7E28		3074	SHLD OUTBUF
0E66	C9		3075	RET
			3076	
			3077	
			3078	*****

Error Addr	Code	Seq	Source statement
		3079	:*****
		3080	:*****
		3081	:
		3082	: BUILD_PREP
		3083	:
		3084	: This routine primes flags and counts, sets up the Last Message sent Type
		3085	: flag, and clears out the Xmit CRC.
		3086	:
		3087	: REGISTERS: A= General use
		3088	:
		3089	: CALLED BY: BUILD
		3090	: SNDNAK
		3091	:
		3092	: FLAGS AND COUNTERS:
		3093	:
		3094	: (CLEARED) XCRC-
		3095	: XCRC+1- The Xmit CRC bytes
		3096	:
		3097	: CRCCN0- Header CRC count
		3098	:
		3099	: CRCCN1- Message CRC count
		3100	:
		3101	: MSGFLG- Flag checked in SNDMSG that indicates
		3102	: whether or not a MSG TYPE is to be sent
		3103	:
		3104	: CTL_COUNT- Count for number of CTL bytes
		3105	:
		3106	: OUTCNT-
		3107	:
		3108	: RDSPER- RD Service Protocol error flag
		3109	:
		3110	: (SET) MSG_TO_SEND- Indicates Message To Send
		3111	:
		3112	: (WRITTEN) HDCNT- Header Count=6
		3113	:
		3114	:
		3115	:
		3116	:
0E67		3117	BUILD_PREP:
0E67	32 7E2B	3118	STA LAST_TYPE_SENT ;Set up the LSTTYP flag
0E6A	AF	3119	XRA A ;Make 0
0E6B	32 7E3B	3120	STA RDSPER ;Make sure this is no longer set
0E6E	32 7E32	3121	STA XCRC ;Zero this out
0E71	32 7E33	3122	STA XCRC+1 ;
0E74	32 7E34	3123	STA CRCCN0 ;clear this
0E77	32 7E35	3124	STA CRCCN1 ;And this
0E7A	32 7E76	3125	STA MSGFLG ;and this
0E7D	32 7E63	3126	STA CTL_COUNT ;Clear this count
0E80	32 7E27	3127	STA OUTCNT ;and this
0E83	3C	3128	INR A ;Make a 1

15-Mar-83 15:22:25

Error	Addr	Code	Seq	Source statement
	0E84	32 7E07	3129	STA MSG_TO_SEND ;Set this flag
	0E87	3E 06	3130	MVI A,6 ;Get a 6
	0E89	32 7E94	3131	STA HDCNT ;Set up this count
	0E8C	C9	3132	RET ;Leave
			3133	;
			3134	;
			3135	;
			3136	;
			3137	;
			3138	;
			3139	;
			3140	;
			3141	;
			3142	;
			3143	;
			3144	;
			3145	;
			3146	;
			3147	;
			3148	;
			3149	;
		=0FFF	3150	ORG 0FFFH ;
			3151	;
	0FFF	00	3152	MIDSUM: DB 0 ;
			3153	;
			3154	;
			3155	;
			3156	;
			3157	;
			3158	;
			3159	;
			3160	;
			3161	;
			3162	;
			3163	;
			3164	ORG 1000H
		=1000	3165	;
	1000		3166	RD_PATTERN:
	1000	A5	3167	DB 0A5H ;Pattern for determining if RD Roms present
			3168	;
	1001		3169	UB_VECTOR:
	1001	C3 16B5	3170	JMP UNLOCK_BUFFER ;Vector to get the the Unlock Input Buffer
			3171	;routine.
			3172	;
	1004		3173	FA_VECTOR:
	1004	C3 101F	3174	JMP FIX_ADDRESS ;
			3175	;
	1007		3176	MOD0_VECTOR:
	1007	C3 1024	3177	JMP MOD0 ;
			3178	;

Error	Addr	Code	Seq	Source statement
	100A		3179	RDD_VECTOR:
	100A	C3 1314	3180	JMP RD_DECODE ;
			3181	
	100D		3182	FBO_VECTOR:
	100D	C3 15C7	3183	JMP FULBU0 ;
			3184	
	1010		3185	TSS_VECTOR:
	1010	C3 111D	3186	JMP TESTSET_S ;
			3187	
	1013		3188	TSQ_VECTOR:
	1013	C3 1113	3189	JMP TESTSET_Q ;
			3190	
	1016		3191	TSQA_VECTOR:
	1016	C3 1105	3192	JMP TSQA ;
			3193	
	1019		3194	HR_VECT08:
	1019	C3 1019	3195	JMP HR_VECT08 ;
			3196	
	101C		3197	HR_VECT09:
	101C	C3 101C	3198	JMP HR_VECT09 ;
			3199	
	101F		3200	FIX_ADDRESS:
	101F	E1	3201	POP H ;Get the new address
	1020	22 4099	3202	SHLD MODEM_ADDRESS ;Save the new address
	1023	C9	3203	RET ;Leave
			3204	
	1024	AF	3205	MOD0: XRA A ;Make a 0
	1025	32 7E06	3206	STA TRNFLAG ;Establish Trans Mode
	1028	32 7E20	3207	STA SOHFLAG ;and get rid of the SOH found flag.
	102B	32 7E1A	3208	STA DPFLAG ;Zero the Data Portion Flag
	102E	32 7E07	3209	STA MSG_TO_SEND ;Kill the Message to Send flag,
	1031	32 7E1C	3210	STA HEADER ;and the Header flag,
	1034	32 7E71	3211	STA LOCAL_COPY ;Get rid of the Local Copy flag
	1037	32 7E72	3212	STA PARALLEL_CNTRL ;and the Parallel Control flag as well.
	103A	32 7E67	3213	STA OVERRIDE_LCS ;and the Override Local Copy in Secure.
	103D	CD 1682	3214	CALL IP_SUB ;Go zap some more things.
	1040	21 0190	3215	LXI H,400 ;Set up a 2 second count
	1043	22 40DA	3216	SHLD DISCNT ;for the Disconnect Count
	1046	3C	3217	INR A ;Make a 1
	1047	32 40D9	3218	STA MIPFLAG ;Set the Modem in Progress flag
			3219	;
			3220	;
			3221	;
			3222	;
			3223	;
			3224	MOD1: MVI A,37H ;Set the Modem Control Bits
	104A	3E 37	3225	OUT TRCR ;Send them to the Command Register
	104C	D3 43	3226	XRA A ;Make a 0
	104E	AF	3227	STA TIKTOK ;Clear this flag
	104F	32 7E0D	3228	MOD1A: MVI A,200 ;Set up a 1 second count
	1052	3E C8		

Error	Addr	Code	Seq	Source statement	
1054	32 7E0C	3229		STA	TIMER2 ;Put it in Timer 2
1057	CD 101F	3230		CALL	FIX_ADDRESS ;Go fix up the entry point address
105A	DB 06	3231	MODEM1:	IN	REMOTE ;Get the Remote Switch
105C	1F	3232		RAR	;Put it in the carry
105D	DA 11C9	3233		JC	MODEM9 ;Jump if not
1060	DB 41	3234		IN	TRSR ;Get the remote SR
1062	17	3235		RAL	;Put DSR in the carry
1063	D2 112B	3236		JNC	FLASH ;Jump if not asserted
1066	D3 3F	3237		OUT	SETRD ;Turn on the R/D light
1068	21 16A8	3238		LXI	H,5800 ;29 second count for Timer1
106B	22 7E0A	3239		SHLD	TIMER1 ;Store it
106E	3E 64	3240		MVI	A,100 ;500 millisecond count for Timer2
1070	32 7E0C	3241		STA	TIMER2 ;Store it
1073	CD 101F	3242		CALL	FIX_ADDRESS ;Go save the next state address
1076	CD 10E0	3243	MODEM2:	CALL	REMDSR ;Check the Remote Switch and the DSR signal
1079	3A 7E0C	3244		LDA	TIMER2 ;Get timer 2
107C	A7	3245		ANA	A ;Is it zero?
107D	C2 1140	3246		JNZ	TICK ;Go check for Ticks
1080	CD 101F	3247		CALL	FIX_ADDRESS ;Go update the address
1083	CD 10E0	3248	MODEM3:	CALL	REMDSR ;Go check the Remote Switch and the DSR signal
		3249			;If it returns, REMOTE*DSR is true. DCD is in
		3250			;the carry
1086	D2 1140	3251		JNC	TICK ;Jump if not set
1089	DB 41	3252		IN	TRSR ;Get the Remote USART SR
108B	1F	3253		RAR	;Put XMIT RDY in the carry
108C	D2 1140	3254		JNC	TICK ;Jump if not set (no CTS)
108F	AF	3255		XRA	A ;Make a 0
1090	32 7E04	3256		STA	TALK_FLAG ;Turn off Talk Mode
1093	32 7E05	3257		STA	DISCARD_RD ;Clear the Discard Remote bit
1096	32 7E70	3258		STA	CTL_C_INTR ;and the Control C interrupt flag
1099	32 7E2D	3259		STA	RD_LD_ERR ;and the RD Load Error flag.
109C	3C	3260		INR	A ;Make a 1
109D	32 7E10	3261		STA	RDCON ;Set the RD Connection Flag
10A0	CD 1105	3262		CALL	TSQA ;Go see if we should send ^Q.
10A3	3E 08	3263	MODM3A:	MVI	A,8 ;Count for LOGCNT
10A5	32 7E0F	3264		STA	LOGCNT ;Set it up
10A8	3E 01	3265		MVI	A,1 ;Make a 1
10AA	32 7E98	3266		STA	SOHENQ ;Set the SOH or ENQ needed flag
10AD	21 0008	3267		LXI	H,8 ;Make a word of 8
10B0	22 7E16	3268		SHLD	ICNT ;Store it
10B3	21 7E68	3269		LXI	H,HBUFFER ;Point to the Header buffer
10B6	22 7E18	3270		SHLD	IADDR ;Store in the the Input Address Buffer
10B9	CD 101F	3271	MOD4:	CALL	FIX_ADDRESS ;Go update the re-entry address
10BC	CD 10E0	3272	MODEM4:	CALL	REMDSR ;Go check Remote and DSR
		3273			;Comes back with DCD in carry
10BF	DA 11EA	3274		JC	RD_FLOW ;Jump if asserted
10C2	2A 7E0A	3275	MOD5:	LHLD	TIMER1 ;Get the contents of TIMER1
10C5	22 7E08	3276		SHLD	TIMSAV ;Save it
10C8	2A 40DA	3277		LHLD	DISCNT ;Get the correct Disconnect Count
10CB	22 7E0A	3278		SHLD	TIMER1 ;Store it

Error	Addr	Code	Seq	Source statement	
10CE	CD	101F	3279	CALL	FIX_ADDRESS
10D1	CD	10E0	3280	MODEMS: CALL	REMDSR
			3281		
10D4	D2	1140	3282	JNC	TICK
10D7	2A	7E08	3283	LHLD	TIMSAV
10DA	22	7E0A	3284	SHLD	TIMER1
10DD	C3	10B9	3285	JMP	MOD4
			3286		
			3287		
			3288		
			3289	;.....	
			3290	;	
10E0	DB	06	3291	REMDSR: IN	REMOTE
10E2	1F		3292	RAR	
10E3	DA	115E	3293	JC	DROP_CONNECTION
10E6	DB	41	3294	IN	TRSR
10E8	17		3295	RAL	
10E9	D2	115E	3296	JNC	DROP_CONNECTION
10EC	17		3297	RAL	
			3298		
10ED	C9		3299	RET	
			3300		
			3301		
			3302		
10EE			3303	TEST_DISCARD:	
10EE	3A	7E05	3304	LDA	DISCARD_RO
10F1	A7		3305	ANA	A
10F2	CA	111D	3306	JZ	TESTSET_S
10F5	C3	1113	3307	JMP	TESTSET_Q
			3308		
			3309		
10F8	3A	7E04	3310	TSQB: LDA	TALK_FLAG
10FB	1F		3311	RAR	
10FC	D8		3312	RC	
10FD	3A	7E46	3313	LDA	MSG15_COUNT
1100	A7		3314	ANA	A
1101	C0		3315	RNZ	
1102	C3	1113	3316	JMP	TESTSET_Q
			3317		
1105	3A	7E8E	3318	TSQA: LDA	OUTLOC
1108	A7		3319	ANA	A
1109	CA	1113	3320	JZ	TESTSET_Q
110C	2A	7E21	3321	LHLD	ACTCNT
110F	7E		3322	MOV	A,M
1110	FE	38	3323	CPI	38H
1112	D0		3324	RNC	
1113			3325	TESTSET_Q:	
1113	21	40D7	3326	LXI	H,SPEND
1116	7E		3327	MOV	A,M
1117	A7		3328	ANA	A

;Point to the next part
 ;Go test Remote and DSR
 ;Comes back with DCD in carry
 ;Go tick the timers as necessary
 ;Get the saved timer count value
 ;Restore TIMER1
 ;Go fix up the MODEM address

;Get the Remote Switch
 ;Put it in the carry
 ;Jump if now in Local
 ;Get the Remote USART SR
 ;Put the DSR in the carry
 ;Jump if not set
 ;Do this to get DCD into carry for the
 ;calling routine
 ;Leave

;Is the Discard Remote Output flag set?
 ;Test it
 ;IF NOT Discard_RO, THEN jump.
 ;ELSE, see about sending ^Q.

;Get the Talk Flag.
 ;Are we in Talk Mode?
 ;IF TALK, THEN leave.
 ;Get this count
 ;Are we doing a message type 15?
 ;IF MSG TYP IS ACTIVE, THEN leave.
 ;ELSE, go see about sending an ^Q.

;Is there an output buffer locked.
 ;Check it.
 ;IF NOT LOCKED, THEN see about sending ^Q.
 ;Get the Address of the Active Count.
 ;Get the count.
 ;Is it >= 38H?
 ;IF >= 38H, THEN leave (don't send ^Q).

;Point to the Control S Flags.
 ;Get them.
 ;Are any set?

Error	Addr	Code	Seq	Source statement	
1118	C8		3329	RZ	;IF NONE SET, THEN leave.
1119	23		3330	INX H	;Point to the Control Q Flags.
111A	36 01		3331	MVI M,1	;Set the Control Q Pending flag.
111C	C9		3332	RET	;THEN leave.
			3333		
111D			3334	TESTSET_S:	
111D	21 40D8		3335	LXI H,QPEND	;Point to the Control Q Flags.
1120	36 00		3336	MVI M,0	;Zap them.
1122	2B		3337	DCX H	;Now point to the Control S Flags.
1123	7E		3338	MOV A,M	;Get them.
1124	A7		3339	ANA A	;Are any set?
1125	C0		3340	RNZ	;IF ANY SET, THEN leave.
1126	3A 40D3		3341	LDA RUN_FLAG	;Get the Run Flag (indicates PM).
1129	77		3342	MOV M,A	;Set or clear the SPEND flag depending on the
			3343		;state of the Run Flag.
112A	C9		3344	RET	;Leave.
			3345	:	
			3346	;	
			3347		
			3348		
112B	CD 1148		3349	FLASH: CALL TICK1	;Go tick the counters
112E	3A 7E0C		3350	LDA TIMER2	;Get the count
1131	A7		3351	ANA A	;Check for 0
1132	CA 1052		3352	JZ MOD1A	;If at 0, go start over
1135	FE 64		3353	CPI 100	;See if we are half way there
1137	D2 113D		3354	JNC 1\$	;Jump if in high half of count
113A	D3 3E		3355	OUT CLRDR	;Turn off the light
113C	C9		3356	RET	;Leave
113D	D3 3F		3357	1\$: OUT SETRD	;Turn the light on
113F	C9		3358	RET	;Leave
			3359		
			3360		
1140	2A 7E0A		3361	TICK: LHLD TIMER1	;Get the 29 second timer
1143	7C		3362	MOV A,H	;Get the high byte
1144	B5		3363	ORA L	;OR in the low byte
1145	CC 115E		3364	CZ DROP_CONNECTION	;Leave if 0
1148	21 7E0D		3365	TICK1: LXI H,TICKTOK	;Point to the Clock State Flag
114B	DB 06		3366	IN SLWCLK	;Get the present clock state
114D	E6 80		3367	ANI 80H	;Save only the MSB
114F	BE		3368	CMP M	;Are they the same?
1150	C8		3369	RZ	;Leave if they are
1151	77		3370	MOV M,A	;Change the flag
1152	2A 7E0A		3371	LHLD TIMER1	;Get timer 1
1155	2B		3372	DCX H	;Decrement it
1156	22 7E0A		3373	SHLD TIMER1	;Put it back
1159	21 7E0C		3374	LXI H,TIMER2	;Point to Timer 2
115C	35		3375	DCR M	;Decrement it
115D	C9		3376	RET	;Leave
			3377		
			3378		

Error	Addr	Code	Seq	Source statement	
115E			3379	DROP_CONNECTION:	
115E	E1		3380	POP H	;Fix up the stack
115F	3A 7E10		3381	LDA RDCON	;Were we connected?
1162	A7		3382	ANA A	;Test the flag
1163	C2 116C		3383	JNZ 1\$	;IF CONNECTED, THEN jump.
1166	CD 10EE		3384	CALL TEST_DISCARD	;ELSE, check the Discard Remote Output situation
1169	C3 117A		3385	JMP DROP_CONO	;THEN go finish dropping the line.
116C	21 16D5		3386	1\$: LXI H,CONLOST_MSG	;Point to this message.
116F	7E		3387	MOV A,M	;Get the count which is in the first byte
1170	32 7E49		3388	STA CLMSG_COUNT	;Store it.
1173	23		3389	INX H	;Point to the text.
1174	22 7E47		3390	SHLD CLMSG_PTR	;Store the address.
1177	CD 111D		3391	CALL TESTSET_S	;Go test and (if necessary) set the S flag.
117A			3392	DROP_CONO:	
117A	21 7E04		3393	LXI H,TALK_FLAG	;Point to the Talk Flag
117D	7E		3394	MOV A,M	;Get it
117E	36 00		3395	MVI M,0	;Zero it out
1180	AF		3396	XRA A	;Make a 0
1181	32 7E10		3397	STA RDCON	;Zero the Connect flag
1184	32 7E0E		3398	STA LOGCON	;Zero the Logical Connection flag
1187	D3 3E		3399	OUT CLRRD	;Turn off the R/D light
1189	3E 35		3400	MVI A,35H	;Set up to clear DTR
118B	D3 43		3401	OUT TRCR	;Do it
118D	21 0190		3402	LXI H,400	;2 second count
1190	22 7E0A		3403	SHLD TIMER1	;Store it
1193	3E 2C		3404	MVI A,44	;220 millisecond count
1195	32 7E0C		3405	STA TIMER2	;Store it
1198	CD 101F		3406	CALL FIX_ADDRESS	;Save the address of the next part
119B	3A 7E0C		3407	MODEM6: LDA TIMER2	;Get Timer #2
119E	A7		3408	ANA A	;Check for 0
119F	C2 1148		3409	JNZ TICK1	;Go check the clock
11A2	CD 101F		3410	CALL FIX_ADDRESS	;Point to the next part
11A5	DB 41		3411	MODEM7: IN TRSR	;Get the SR
11A7	17		3412	RAL	;Check DSR
11A8	D2 11B3		3413	JNC 1\$	;Jump if not asserted
11AB	2A 7E0A		3414	LHLD TIMER1	;Get the 2 second timer
11AE	7C		3415	MOV A,H	;Get the high byte
11AF	B5		3416	ORA L	;OR in the low
11B0	C2 1148		3417	JNZ TICK1	;If not zero, go tick the timers
11B3	CD 101F		3418	1\$: CALL FIX_ADDRESS	;Go fix up the Modem Pointer.
11B6	3A 7E49		3419	MODEM8: LDA CLMSG_COUNT	;Get the Connection Lost Message Count.
11B9	21 7E46		3420	LXI H,MSG15_COUNT	;Point to the Message Type 15 Count.
11BC	B6		3421	ORA M	;Are they both zero?
11BD	C2 0809		3422	JNZ STT_VECTOR	;If not zero, go send the Transparent Text.
11C0	CD 10EE		3423	CALL TEST_DISCARD	;Go test the Discard Remote Output situation.
11C3	DB 06		3424	IN REMOTE	;Get the keyswitch position
11C5	1F		3425	RAR	;Are we in Remote?
11C6	D2 104A		3426	JNC MOD1	;IF REMOTE, THEN jump.
11C9	AF		3427	MODEM9: XRA A	;Make a 0
11CA	32 7E04		3428	STA TALK_FLAG	;Turn off the Talk Flag

Error	Addr	Code	Seq	Source statement	
	11CD	CD 1113	3429	CALL TESTSET_Q	;Decide about whether or not we have to
			3430		;set the Q Pending flag.
	11D0	3E 15	3431	MVI A,15H	;Clear RTS
	11D2	D3 43	3432	OUT TRCR	;Do it
	11D4	D3 3E	3433	OUT CLRRD	;Make sure the RD light is off
	11D6	21 40BD	3434	LXI H,LR_MASK_WORK	;Point to Working Mask
	11D9	7E	3435	MOV A,M	;Get it
	11DA	E6 DB	3436	ANI ODBH	;Clear the remote bits
	11DC	77	3437	MOV M,A	;Store the new version
	11DD	23	3438	INX H	;Bump the pointer twice
	11DE	23	3439	INX H	;
	11DF	7E	3440	MOV A,M	;Get the Mask Copy
	11E0	E6 DB	3441	ANI ODBH	;Clear the remote bits
	11E2	77	3442	MOV M,A	;Restore it
	11E3	AF	3443	XRA A	;Make a 0
	11E4	32 40D9	3444	STA MIPFLG	;Clear the Modem In Progress flag
	11E7	C3 004F	3445	JMP FU_VECTOR	;and go start from scratch
			3446	;	
			3447	;	
			3448		
			3449		
			3450	;	
			3451	;	
	11EA		3452	RD_FLOW:	
	11EA	3A 7E07	3453	LDA MSG_TO_SEND	;Get the Message to Send flag
	11ED	1F	3454	RAR	;Put it in the carry
	11EE	DC 1401	3455	CC SNDMSG	;Go send it
	11F1	3A 7E1F	3456	LDA MSG_TO_READ	;Get the Message to Read Flag
	11F4	1F	3457	RAR	;Put it in the carry
	11F5	DC 14BA	3458	CC MSG_TR_DECODE	;Go figure out what it is.
	11F8	3A 7E02	3459	LDA REMXSR	;Get the Remote Xmit SR
	11FB	1F	3460	RAR	;Is it ready
	11FC	DA 1209	3461	JC 1\$	;Jump if it is (no character waiting)
	11FF	3A 7E06	3462	LDA TRNFLG	;Get the Trans Mode flag
	1202	1F	3463	RAR	;Put it in the carry
	1203	D4 157C	3464	CNC TRNSND	;Go do a Trans Mode send
	1206	DC 158C	3465	CC PROSND	;Otherwise, do a Protocol Mode send
	1209	0E 00	3466	1\$: MVI C,0	;Prime this
	120B	DB 41	3467	IN TRSR	;Get the Remote Status Register
	120D	47	3468	MOV B,A	;Save a copy in B
	120E	E6 02	3469	ANI 2	;Check for RCVR DONE
	1210	CA 1231	3470	JZ 3\$	;Jump if not
	1213	DB 40	3471	IN TRDB	;Get the character right away
	1215	32 7E12	3472	STA RDTEMP	;Save it here
	1218	78	3473	MOV A,B	;Get the SR back
	1219	E6 20	3474	ANI 20H	;Check for Framing Errors
	121B	C2 124D	3475	JNZ CONERR	;Go check the Connection Error situation
	121E	3A 7E20	3476	LDA SOHFLG	;Get the SOH Flag
	1221	1F	3477	RAR	;Is it set?
	1222	DA 122C	3478	JC 2\$	;Skip the Trans Mode stuff if it is.

Error	Addr	Code	Seq	Source statement
			3479	;(Note: We jump with the carry set so the CC
			3480	;at 2\$ will work.)
1225	3A	7E06	3481	LDA TRNFLG ;Get the Trans Mode Flag
1228	1F		3482	RAR ;See if asserted (asserted low)
1229	D4	1349	3483	CNC TRANS_RECEIVE ;Go service it if in Trans Mode
122C	DC	1375	3484	2\$: CC PROTO_RECEIVE ;Otherwise, go to Protocol Receive
122F	0E	FF	3485	MVI C,-1 ;Set up to subtract 1 from LOGCNT
1231	21	7E0F	3486	3\$: LXI H,LOGCNT ;Point to the Logical Connection count
1234	7E		3487	MOV A,M ;Get it
1235	A7		3488	ANA A ;Is it 0?
1236	C2	1248	3489	JNZ 4\$;Jump if it isn't
1239	3A	7E1C	3490	LDA HEADER ;Get the Header Received Flag
123C	1F		3491	RAR ;Is it set?
123D	DC	125A	3492	CC HDECOD ;Go decode it
1240	3A	7E1D	3493	LDA MESSAG ;Get the Message Received Flag
1243	1F		3494	RAR ;Is it set?
1244	DC	0812	3495	CC MD_VECTOR ;If set, go decode the message
1247	C9		3496	RET ;Leave
			3497	
1248	81		3498	4\$: ADD C ;Otherwise, add the contents of C
1249	77		3499	MOV M,A ;Store the result
124A	C3	1140	3500	JMP TICK ;Go check the timers
			3501	
			3502	
124D	3E	37	3503	CONERR: MVI A,37H ;Prepare to reset the error
124F	D3	43	3504	OUT TRCR ;Do it
1251	21	7E0F	3505	LXI H,LOGCNT ;Point to the Logical Connection indicator
1254	7E		3506	MOV A,M ;Get the count
1255	A7		3507	ANA A ;Is it 0?
1256	C8		3508	RZ ;Leave if it is
1257	36	08	3509	MVI M,8 ;Initialize it back to 8 if not 0
1259	C9		3510	RET ;And leave
			3511	;
			3512	;.....
			3513	
			3514	
			3515	
			3516	;.....
			3517	;.....
			3518	;
			3519	; HDECOD
			3520	; ENQDEC
			3521	;
			3522	; Header Decode routine. When a complete header has been received, it is
			3523	; subsequently decoded by this routine. The first byte is checked for SOH
			3524	; or ENQ. If SOH, then we jump to SOH_DECODE. If ENQ, we fall through
			3525	; and figure out the type of ENQ: ACK, NACK, or REP.
			3526	;
			3527	; REGISTERS: A= General use
			3528	; B= Message type code (1=ACK, 2=NACK)

Error	Addr	Code	Seq	Source statement	
			3529	:	C= CTL REASON code
			3530	:	
			3531	:	ENTERED FROM: RD3
			3532	:	
			3533	:	FLAGS AND COUNTERS:
			3534	:	
			3535	:	(READ)
			3536	:	HCOUNT- (aka CTL TYPE and CTL reason)
			3537	:	HBUFFER- Header type (SOH or ENQ)
			3538	:	ENQHED+2- CTL REASON from last ENQ sent
			3539	:	LAST_TYPE_SENT- Last message type sent
			3540	:	
125A	2A	7E69	3541	HDECOD: LHLD HCOUNT	;Get the Count (aka CTL TYPE and CTL REASON)
125D	EB		3542	XCHG	;Put it in DE
125E	3A	7E68	3543	LDA HBUFFER	;Get the received Message Type
1261	FE	81	3544	CPI 81H	;Is it an SOH?
1263	CA	12BF	3545	JZ SOH_DECODE	;Go to SOH Decode if it is
1266	7B		3546	ENQDEC: MOV A,E	;Get the CTL TYPE
1267	FE	01	3547	CPI 1	;See if ACK
1269	CA	1286	3548	JZ RDACK	;Jump if ACK
126C	FE	02	3549	CPI 2	;Check for NACK
126E	CA	12B8	3550	JZ RDNACK	;Jump if NACK
1271	CD	1298	3551	RDREP: CALL INFIX	;Go set up the Input Buffer flags
1274	3A	7EA9	3552	LDA ENQHED+2	;Get the CTL REASON from the last ENQ sent
1277	4F		3553	MOV C,A	;Put it in C
1278	3A	7E2B	3554	LDA LAST_TYPE_SENT	;Get my Last Message Type flag
127B	FE	02	3555	CPI 2	;Check for less than 2
127D	D2	1606	3556	JNC SNDNAK	;Go set up to send a NACK
1280	01	0100	3557	LXI B,100H	;Set up a 1 in B and a 0 in C
1283	C3	1608	3558	JMP SNDNA0	;Go use part of the SNDNAK flow to set up for
			3559		;sending an ACK
			3560		
			3561		
1286	CD	1298	3562	RDACK: CALL INFIX	;Go do the Input Header stuff.
1289	21	7E90	3563	RDACK1: LXI H,LBSENT	;Point to the Locked Buffer Sent flag
128C	7E		3564	MOV A,M	;Get it
128D	A7		3565	ANA A	;Is it set?
128E	C8		3566	RZ	;Leave if not
128F	AF		3567	XRA A	;Make a 0
1290	77		3568	MOV M,A	;Zero it now
1291	2B		3569	DCX H	;Point to the LBSIP flag
1292	77		3570	MOV M,A	;and zero it
1293	2B		3571	DCX H	;Point to the Locked Flag
1294	77		3572	MOV M,A	;and zero it.
1295	C3	10F8	3573	JMP TSQB	;Go see about sending ^Q.
			3574		
1298	11	0006	3575	INFIX: LXI D,6	;Set up a count of 6
129B	21	7E68	3576	LXI H,HBUFFER	;Point to the Header Buffer.
129E	22	7E18	3577	INFIX0: SHLD IADDR	;Save the address
12A1	22	7E42	3578	SHLD IBUF_ADR_COPY	;Save another copy

Error Addr	Code	Seq	Source statement
12A4	EB	3579	XCHG ;Get the count into HL
12A5	22 7E40	3580	SHLD IBUF_CNT_COPY ;Save a copy of the count here
12A8	23	3581	INX H ;Do this twice to add two
12A9	23	3582	INX H ;to account for the CRC.
12AA	22 7E16	3583	SHLD ICNT ;Store it here for use by PROTO_RECEIVE
12AD	AF	3584	XRA A ;Make a 0
12AE	32 7E1C	3585	STA HEADER ;Clear the Header flag
12B1	32 7E30	3586	STA RCRC ;Clear the CRC for the receive side
12B4	32 7E31	3587	STA RCRC+1 ;
12B7	C9	3588	RET ;Leave
		3589	
12B8	AF	3590	RDNACK: XPA A ;Make a 0 for LAST_TYPE_SENT code
12B9	CD 0818	3591	CALL BUILD_VECTOR ;Go clean up the SNDMSG counts, pointers, etc.
12BC	C3 1298	3592	JMP INFIX ;Go build the input stuff.
		3593	;
		3594	;
		3595	;
		3596	;
		3597	;
		3598	;
		3599	;
		3600	;
		3601	;
		3602	;
		3603	;
		3604	;
		3605	;
		3606	;
		3607	;
		3608	;
		3609	;
		3610	;
		3611	SOH_DECODE:
12BF	AF	3612	XRA A ;Make a 0
12C0	32 7E36	3613	STA NEEDI ;Zero the Need Init flag
12C3	21 7E6B	3614	LXI H,LMSGNM ;Point to the Last Message Number field
12C6	7E	3615	MOV A,M ;Get it
12C7	23	3616	INX H ;Point to the Current Message Number field
12C8	BE	3617	CMP M ;See if they are the same
12C9	C2 1604	3618	JNZ NACK7 ;Leave if not
12CC	21 7E2A	3619	LXI H,LAST_GOOD_MSG ;Point to my Last Good Data Message received
		3620	number.
12CF	46	3621	MOV B,M ;Put it in B
12D0	04	3622	INR B ;Bump it by one
12D1	B8	3623	CMP B ;Are they the same?
12D2	CA 12DD	3624	JZ 1\$;If they are, all is well so skip next part
12D5	A7	3625	ANA A ;Check for 0 in LMSGNM and CMSGNM
12D6	C2 1604	3626	JNZ NACK7 ;Error if not
12D9	3C	3627	INR A ;Make a 1
12DA	32 7E36	3628	STA NEEDI ;Set the Need Init flag

Error	Addr	Code	Seq	Source statement	
	12DD	7A	3629	1\$: MOV A,D	;Get the high byte of the count
	12DE	A7	3630	ANA A	;Must be zero
	12DF	C2 15FF	3631	JNZ NACK6	;Jump if message too long
	12E2	7B	3632	MOV A,E	;Get the low byte of the count.
	12F3	FE 82	3633	CPI 130	;Check for count>input buffer
	12E5	D2 15FF	3634	JNC NACK6	;Jump if count greater than 130
	12E8	3A 7E1B	3635	LDA INLOCK	;Get the Input Buffer Locked Flag
	12EB	A7	3636	ANA A	;Is it zero?
	12EC	CA 12FB	3637	JZ 2\$	;Jump if it is
	12EF	7B	3638	MOV A,E	;Get the low byte of the count.
	12F0	FE 02	3639	CPI 2	;See if it is 0 or 1
	12F2	D2 15F5	3640	JNC NACK4	;Error if not
	12F5	21 7E60	3641	LXI H,IBUF1	;Point to the short data buffer
	12F8	C3 12FF	3642	JMP 3\$	;Jump with a 1 in A
	12FB	21 7EDB	3643	2\$: LXI H,IBUF0	;Point to the long Data buffer
	12FE	AF	3644	XRA A	;Make a 0 to indicate Buffer 0
	12FF	32 7E1E	3645	3\$: STA INFLAG	;Indicate which Input buffer is being used
	1302	3E 01	3646	MVI A,1	;Make a 1
	1304	32 7E1A	3647	STA DPFLAG	;Set the Data Portion Flag
	1307	32 7E06	3648	STA TRNFLG	;Set this to Protocol Mode
	130A	CD 129E	3649	CALL INFIX0	;Go build the input stuff.
	130D	AF	3650	XRA A	;Make a 0
	130E	32 7E20	3651	STA SOHFLG	;and clear the SOH Flag.
	1311	C3 1289	3652	JMP RDACK1	;Go share the cleanup code.
			3653	:	
			3654	;	*****
			3655	;	*****
			3656	:	
			3657	:	
			3658	;	*****
			3659		
			3660		
			3661		
			3662		
			3663		
			3664	;	*****
			3665	;	*****
			3666	:	
			3667	:	
			3668	RD_DECODE:	
1314			3669	LHLD IBUF_CNT_COPY	;Get the count
1314	2A 7E40		3670	MOV B,L	;Put it in B
1317	45		3671	LHLD IBUF_ADR_COPY	;Get the address of the correct buffer
1318	2A 7E42		3672	MOV A,M	;Get the first byte
1318	7E		3673	ANI 20H	;Save the CTL bit
131C	E6 20		3674	STA CTLFLG	;Set or clear this flag
131E	32 7E97		3675	MOV A,M	;Get the first byte again
1321	7E		3676	ANI 1FH	;Save only the low 5 bits
1322	E6 1F		3677	MOV C,A	;Save the MSG TYPE field in C
1324	4F		3678	LDA NEEDI	;Do we want only an INIT?
1325	3A 7E36				

Error	Addr	Code	Seq	Source statement
	1328	A7	3679	ANA A ;Check the flag
	1329	79	3680	MOV A,C ;Get the MSG TYPE back into A
	132A	CA 1336	3681	JZ 1\$;Jump if NEEDI=0
	132D	FE 01	3682	CPI 1 ;See if it is indeed an Init Protocol msg
	132F	CA 1336	3683	JZ 1\$;IF INIT PROTOCCL, THEN jump.
	1332	E1	3684	POP H ;Fix up the stack because we don't want to go
			3685	;back to MSG_DECODE after we complete the NACK
			3686	;preparations, but one level higher instead.
	1333	C3 1604	3687	JMP NACK7 ;Error if not
	1336	FE 10	3688	1\$: CPI 16 ;Make sure the number is within range
	1338	D2 1674	3689	JNC RESERVED ;Jump if not
	133B	3F	3690	CMC ;Clear the carry
	133C	17	3691	RAL ;Multiply by two
	133D	5F	3692	MOV E,A ;Put the multiplied value in E
	133E	16 00	3693	MVI D,0 ;Clear the high byte
	1340	21 1654	3694	LXI H, RD_COMMAND ;Point to the RD Command Table
	1343	19	3695	DAD D ;Add in the multiplied value
	1344	5E	3696	MOV E,M ;Get the first byte out of the table
	1345	23	3697	INX H ;Point to the next
	1346	56	3698	MOV D,M ;Get it
	1347	EB	3699	XCHG ;Put the address in HL
	1348	E9	3700	PCHL ;Go there
			3701	;
			3702	;
			3703	END OF RD_DECODE ROUTINE.
			3704	;
			3705	*****
			3706	*****
			3707	*****
			3708	*****
			3709	*****
			3710	*****
			3711	;
			3712	;
			3713	TRANS_RECEIVE
			3714	;
			3715	Transparent Mode Receive routine. This is the routine that picks up
			3716	characters when the RD is in Trans Mode. However, if the SOH Flag is
			3717	set, it is assumed that we are in the process of picking up a header
			3718	and are thus preparing to switch from Transparent Mode to Protocol Mode.
			3719	;
			3720	If the SOHFLG is not set, we must see if the character is an SOH (81H).
			3721	If it is, we set the SOHFLG and leave
			3722	;
			3723	;
			3724	REGISTERS: A= General use.
			3725	HL= Pointer to REMRDB and REMRSR
			3726	;
			3727	;
			3728	CALLLED BY: RD
				;
				FLAGS AND COUNTERS:

Error	Addr	Code	Seq	Source statement
			3729	;
			3730	;
			3731	(CLEARED) Carry- Inhibits call to PROTO_RECEIVE after
			3732	successful completion of TRANS_RECEIVE
			3733	;
			3734	(SET) SOHFLG- Set if SOH detected
			3735	;
			3736	(READ) SOHFLG- Indicates SOH found and header reception
			3737	in progress.
			3738	;
			3739	(WRITTEN) REMRSR- Receiver Done is set if -(SOH+SOHFLG).
			3740	Overflow is set if Receiver Done already
			3741	set.
			3742	;
1349			3743	TRANS_RECEIVE:
1349	3A	7E20	3744	LDA SOHFLG ;Get the SOH found flag
134C	1F		3745	RAR ;Is it set? (NOTE: This sets the carry for the
			3746	;PROTO_RECEIVE check
134D	D8		3747	RC ;Leave if set
134E	3A	7E12	3748	LDA RDTEMP ;Get the character
1351	FE	81	3749	CPI 81H ;Check for SOH
1353	CA	1368	3750	JZ SOHFND ;Jump if it is
1356			3751	TRANS_RECV:
1356	21	7E01	3752	LXI H, REMRDB ;Point to the Remote DB
1359	77		3753	MOV M, A ;Store the character
135A	2B		3754	DCX H ;Back up the pointer
135B	7E		3755	MOV A, M ;Get the SR
135C	E6	02	3756	ANI 2 ;Check it
135E	CA	1363	3757	JZ 1\$;Jump if Done not already set
1361	3E	10	3758	MVI A, 10H ;Set the overrun bit
1363	F6	02	3759	1\$: ORI 2 ;Set the REMOTE DONE flag
1365	77		3760	MOV M, A ;Store the SR
1366	AF		3761	XRA A ;Clear the carry to make keep us from going
			3762	;to PROTO_RECEIVE after we leave TRANS_RECEIVE
1367	C9		3763	RET ;Leave
			3764	
			3765	
			3766	*****
			3767	*****
			3768	;
			3769	SOHFND
			3770	;
			3771	We come here from TRANS_RECEIVE if the received character is an SOH.
			3772	We clear out the Receive CRC word and set the SOHFLG to indicate that
			3773	an SOH was detected. Also we set the carry so PROTO_RECEIVE will be
			3774	called after we leave this flow.
			3775	;
			3776	;
			3777	REGISTERS: A= General use
			3778	HL= Holds a word of zero.

Error Addr	Code	Seq	Source statement
------------	------	-----	------------------

		3779	:
		3780	:
		3781	:
		3782	:
		3783	:
		3784	:
		3785	:
		3786	:
		3787	:
		3788	:
		3789	:
		3790	:
		3791	:
		3792	:
		3793	:
		3794	:
		3795	:
		3796	:
		3797	:
		3798	:
		3799	:
		3800	:
		3801	:
		3802	:
		3803	:
		3804	:
		3805	:
		3806	:
		3807	:
		3808	:
		3809	:
		3810	:
		3811	:
		3812	:
		3813	:
		3814	:
		3815	:
		3816	:
		3817	:
		3818	:
		3819	:
		3820	:
		3821	:
		3822	:
		3823	:
		3824	:
		3825	:
		3826	:
		3827	:
		3828	:

ENTERED FROM: TRANS_RECEIVE

FLAGS AND COUNTERS:

(CLEARED)

RCRC-
RCRC+1- The Receive CRC is primed.

(SET)

SOHFLG- Indicates SOH has been detected.

Carry- Causes PROTO_RECEIVE to be entered
after we leave this flow.

1368	21	0000
136B	22	7E30
136E	3E	01
1370	32	7E20
1373	37	
1374	C9	

SOHFND:	LXI	H,0	;Make a word of 0
	SHLD	RCRC	;Zero out the Receive CRC
	MVI	A,1	;Make a 1
	STA	SOHFLG	;Set this flag
	STC		;Set the carry
	RET		;Leave

;Remember to only take characters out of the buffer if the REMRSR done bit is
;cleared.

;NOTE: If CRC Error * Trans Mode, set up ICNT, IADDR and SOHENQ.

PROTO_RECEIVE

PROTO_RECEIVE:

LXI	H,SOHENQ	;Get the SOH or ENQ needed flag
MOV	A,M	;Get it
ANA	A	;Set or clear the Z bit
LDA	RDTEMP	;Get the character
JZ	2\$	;Jump if the SOHENQ flag is clear
CPI	81H	;Is the character an SOH?
JZ	1\$	;If it is, go clear the SOHENQ flag
CPI	5	;Is it an ENQ?
RNZ		;Leave if not

1375	21	7E98
1375	7E	
1378	A7	
1379	3A	7E12
137A	CA	138A
137D	FE	81
1380	CA	1388
1382	FE	05
1385	C0	
1387		

Error	Addr	Code	Seq	Source statement
	1388	36 00	3829	1\$: MVI M,0 ;Zero out the SOH or ENQ needed flag
	138A	CD 13D4	3830	2\$: CALL RCVCRC ;Go CRC it
	138D	2A 7E18	3831	LHLD IADDR ;Get the Address for storing the character
	1390	77	3832	MOV M,A ;Store it
	1391	23	3833	INX H ;Bump the address
	1392	22 7E18	3834	SHLD IADDR ;Save the updated address
	1395	2A 7E16	3835	LHLD ICNT ;Get the Protocol Input Count
	1398	2B	3836	DCX H ;Subtract one
	1399	22 7E16	3837	SHLD ICNT ;Save the new count
	139C	7D	3838	MOV A,L ;Put the low byte in A
	139D	B4	3839	ORA H ;OR in the high byte
	139E	C0	3840	RNZ ;Leave if not zero
	139F	21 7E30	3841	LXI H,RCRC ;Point to the CRC
	13A2	7E	3842	MOV A,M ;Get the low byte
	13A3	23	3843	INX H ;Point to the high byte
	13A4	B6	3844	ORA M ;OR it in
	13A5	C2 15ED	3845	JNZ NACK1 ;Jump if not 0
	13A8	3A 7E1A	3846	LDA DPFLAG ;See if we are doing a Data Portion
	13AB	A7	3847	ANA A ;Is the DP Flag set?
	13AC	C2 13B4	3848	JNZ 3\$;Jump if in Data Portion
	13AF	3C	3849	INR A ;Make a 1
	13B0	32 7E1C	3850	STA HEADER ;Set the Header Found flag
	13B3	C9	3851	RET ;Leave
	13B4	32 7E1D	3852	3\$: STA MESSAG ;Set the Message Found flag
	13B7	3A 7E1E	3853	LDA INFLAG ;Get the Input Buffer in use flag.
	13BA	A7	3854	ANA A ;Is it 0 (Long Buffer) or 1 (Short Buffer)?
	13BB	C2 13C2	3855	JNZ 4\$;IF SHORT BUFFER, THEN jump.
	13BE	3C	3856	INR A ;ELSE, make a 1.
	13BF	32 7E1B	3857	STA INLOCK ;Lock the input buffer
	13C2	3A 7E36	3858	4\$: LDA NEEDI ;See if this is a NEED INITPROTOCOL situation.
	13C5	A7	3859	ANA A ;Is the flag set?
	13C6	C0	3860	RNZ ;IF SET, THEN leave.
	13C7	3A 7E6C	3861	LDA MSGNM ;ELSE, get the Message number from the header
	13CA	32 7E2A	3862	STA LAST_GOOD_MSG ;Set up the Last Good Message number
	13CD	C9	3863	RET ;and leave
			3864	;
			3865	;.....
			3866	;
			3867	;
			3868	;
			3869	;.....
			3870	;
			3871	;
			3872	;
			3873	;
			3874	;
			3875	;
			3876	;
			3877	;
			3878	;

XMTCRC
 RCVCRC
 CRCSUB

 This is where the CRC for both the Transmit and Receive sides is generated. The only difference is in the pointer in HL.

Error Addr Code

Seq Source statement

```
3879 ; REGISTERS: A= General use
3880 ; B= Another general use register
3881 ; C= Loop count
3882 ; D= Work register holding the high byte of CRC
3883 ; E= Work register holding the low byte of CRC
3884 ; HL= Point to the CRC word
3885 ;
3886 ; ENTERED FROM: SNDMSG (XMTCRC)
3887 ; SNDMSA (XMTCRC)
3888 ; SNDMS3 (XMTCRC)
3889 ;
3890 ; PRORC1 (RCVCRC)
3891 ;
3892 ; STATE SAVED: The character and the PSW are saved on the stack
3893 ; and restored before we leave.
3894 ;
3895 ;
3896 ;
3897 ; #####
3898 ;
3899 ; EXPLANATION OF FLOW: First we set up the correct pointer in HL.
3900 ; Then, both XMTCRC and RCVCRC enter the CRC SUB flow. Here the presently
3901 ; accumulated CRC is transferred into DE, the character is put in B,
3902 ; and a count of 8 is loaded into C. Also the character is saved on the stack.
3903 ; Now we do the next five steps eight times.
3904 ;
3905 ; DO EIGHT(8) TIMES:
3906 ;
3907 ; XOR section:
3908 ;
3909 ; 1. R(B)00 <-- R(B)00 [XOR] CRC00
3910 ; 2. CRC01 <-- R(B)00 [XOR] CRC01
3911 ; 3. CRC14 <-- R(B)00 [XOR] CRC14
3912 ;
3913 ; Note: R(B)00 in steps 2 and 3 is the R(B)00 after step 1
3914 ;
3915 ; SHIFT section:
3916 ;
3917 ; 4. R(B) <-- R(B) [right shifted once with the
3918 ; LSB going into the Carry]
3919 ; 5. CRC <-- CRC [right shifted once with the
3920 ; MSB coming from the carry and the
3921 ; LSB going into the Bit Bucket.]
3922 ;
3923 ; Each time through the loop we work on the next bit in the character because
3924 ; the next significant bit gets shifted into R(B) bit 00 at step 4 above.
3925 ;
3926 ; #####
3927 ;
3928 ;
```

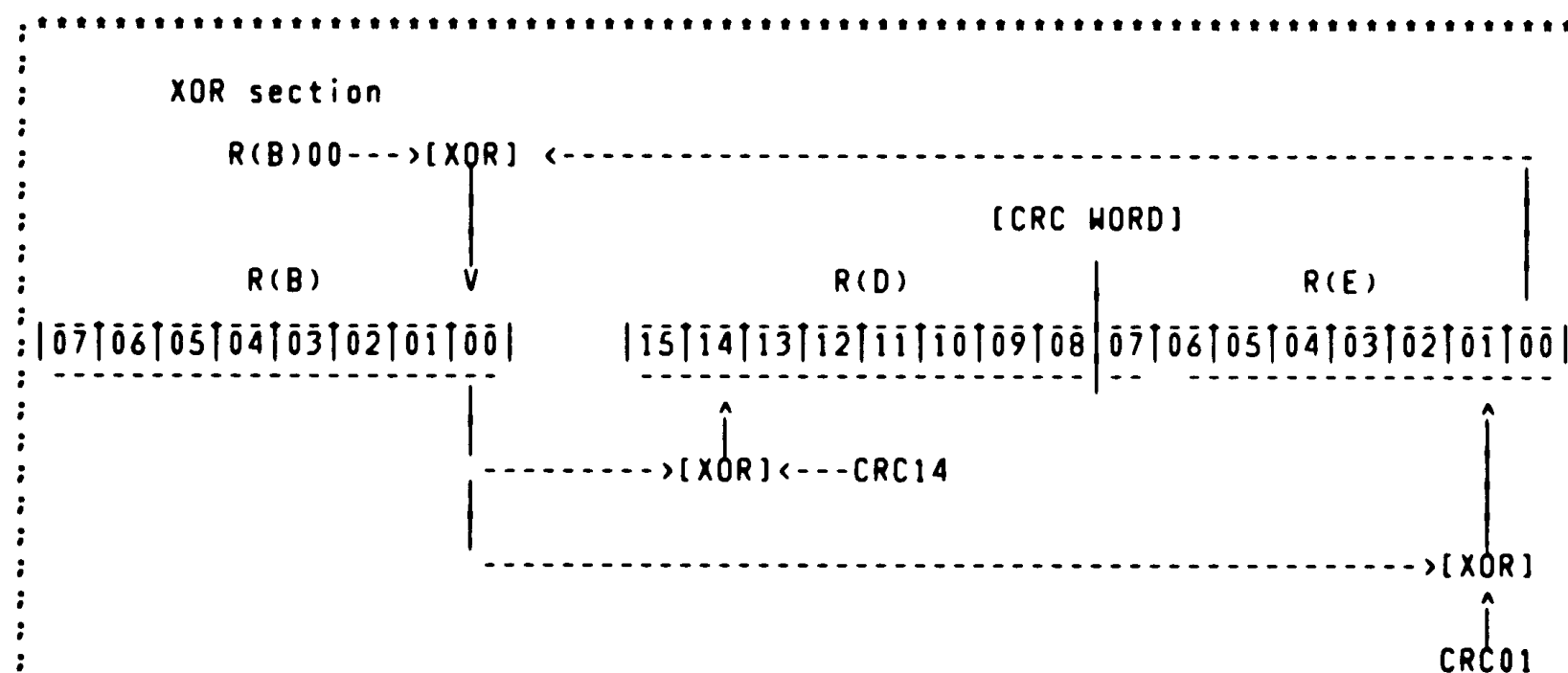
Error Addr	Code	Seq	Source statement
------------	------	-----	------------------

13CE	21 7E32	3929	XMTCRC: LXI H,XCRC ;Point to this CRC
13D1	C3 13D7	3930	JMP CRCSUB ;Go share the actual CRC routine

13D4	21 7E30	3931	
		3932	RCVCRC: LXI H,RCRC ;Point to this CRC word

		3933	
		3934	; Now we fall through
		3935	

13D7	F5	3936	CRCSUB: PUSH PSW ;Save the character
13D8	5E	3937	MOV E,M ;Get the low byte
13D9	23	3938	INX H ;Point to the high byte
13DA	56	3939	MOV D,M ;Get it
13DB	47	3940	MOV B,A ;Put the character in B
13DC	0E 08	3941	MVI C,8 ;Set up a count of 8 in C



13DE	7B	3968	1\$: MOV A,E ;Get the low order CRC byte
13DF	E6 01	3969	ANI 1 ;Isolate the LSB
13E1	A8	3970	XRA B ;XOR the the result with the character
13E2	47	3971	MOV B,A ;Put the result back in B

; Now that we have XORed the LSB of the current CRC value with the current
 ; bit of the character, we can XOR the result with bits 1 and 14 of the CRC.

13E3	E6 01	3976	ANI 1 ;Isolate the LSB
13E5	07	3977	RLC ;Rotate the result into bit 1
13E6	AB	3978	XRA E ;XOR it with the low byte of the CRC

Error	Addr	Code	Seq	Source statement
	13E7	5F	3979	MOV E,A ;Put the byte back
	13E8	78	3980	MOV A,B ;Get the character back
	13E9	E6 01	3981	ANI 1 ;Isolate the LSB again
	13EB	0F	3982	RRC ;Rotate it into bit 6 since this is bit 14
	13EC	0F	3983	RRC ;of the CRC word.
	13ED	AA	3984	XRA D ;XOR it against the high CRC byte
	13EE	57	3985	MOV D,A ;Put back the new version
			3986	:
			3987	:.....
			3988	:
			3989	:
			3990	:
			3991	:.....
			3992	:
			3993	: SHIFT section:
			3994	:
			3995	: CARRY (which equals 0)
			3996	:
			3997	: V R(B) R(D) R(E)
			3998	: 07 06 05 04 03 02 01 00 --> 15 14 13 12 11 10 09 08 07 06 05 04 03 02 01 00
			3999	: ----- ----- -----
			4000	:
			4001	:
			4002	:
			4003	: At this point in the loop, R(B) bit 00 contains the result of XORing the LSB
			4004	: of the current CRC with the current bit we are working on in the character
			4005	: being CRC'd. Note that the original state of that bit has been lost. CRC
			4006	: bit 14 (R(D) bit 06) contains the XOR of bit 14 of the CRC and bit 00 of
			4007	: R(B). CRC bit 01 (R(E) bit 01) contains the XOR of bit 01 of the CRC and
			4008	: bit 00 of R(B).
			4009	:
			4010	: Our next step is to shift everything right once. When this is done we will
			4011	: have put a 0 in R(B) bit 07, R(B) bit 00 will contain the next character bit
			4012	: that we want to work on, the result of XORing the last character bit we
			4013	: worked on with the LSB of the previously accumulated CRC will be shifted
			4014	: from R(B) bit 00 to [CRC bit 15], and the CRC will have been shifted right
			4015	: once.
			4016	:
			4017	:
			4018	:
			4019	:
			4020	: NOTE: This shift puts R(B)00 in the carry and the next bit we want to
			4021	: work on (in the byte we are CRCing) in R(B)00.
			4022	:
	13EF	78	4023	MOV A,B ;Get the character
	13F0	1F	4024	RAR ;MSB of R(B) <-- (CY=0)
			4025	; (CY) <-- LSB of R(B)
	13F1	47	4026	MOV B,A ;Put the result back in B
			4027	:
			4028	:

Error	Addr	Code	Seq	Source statement
			4029	; NOTE: These next two rotates are the shift part of the [XOR and shift]
			4030	; algorithm that generates a CRC.
			4031	
13F2	7A		4032	MOV A,D ;Get the high byte of the CRC
13F3	1F		4033	RAR ;MSB of CRC(HB) <-- (CY=LSB of R(B)
			4034	; (CY) <-- LSB of CRC(HB)
13F4	57		4035	MOV D,A ;Put the result back
			4036	
13F5	7B		4037	MOV A,E ;Get the low byte of the CRC
13F6	1F		4038	RAR ;MSB of CRC(LB) <-- (CY=LSB of CRC(HB))
			4039	; (CY) <-- LSB of CRC(LB), but we don't care
			4040	; about this bit. This is the proverbial Bit
			4041	; Bucket!
13F7	5F		4042	MOV E,A ;Put the result back
			4043	;
			4044	;
			4045	;
13F8	0D		4046	DCR C ;Decrement the loop count
13F9	C2 13DE		4047	JNZ 1\$;Loop back if not done
13FC	72		4048	MOV M,D ;Store the high byte of the CRC
13FD	2B		4049	DCX H ;Back the pointer in HL
13FE	73		4050	MOV M,E ;Store the low byte of the CRC
13FF	F1		4051	POP PSW ;Get the character back
1400	C9		4052	RET ;NOTE: AC must equal character for the return
			4053	;
			4054	;
			4055	;
			4056	;
			4057	;
			4058	;
			4059	;
			4060	;
			4061	;
			4062	;
			4063	SNDMSG: IN TRSR ;Get the Remote SR
1401	DB 41		4064	RAR ;Look for Xmit Ready
1403	1F		4065	RNC ;Leave if not set
1404	D0		4066	LDA HDCNT ;Get the Header Count
1405	3A 7E94		4067	ANA A ;Is it zero?
1408	A7		4068	JZ 1\$;Jump if zero
1409	CA 1424		4069	LHLD HDADR ;Get the correct address
140C	2A 7E95		4070	MOV A,M ;Get the byte to be sent
140F	7E		4071	OUT TRDB ;Send it
1410	D3 40		4072	INX H ;Bump the pointer
1412	23		4073	SHLD HDADR ;Save the address
1413	22 7E95		4074	CALL XMTCRC ;Go CRC it
1416	CD 13CE		4075	LXI H,HDCNT ;Point to this count
1419	21 7E94		4076	DCR M ;Subtract 1
141C	35		4077	RNZ ;Leave if not done
141D	C0		4078	MVI A,2 ;Set up this count
141E	3E 02			

Error	Addr	Code	Seq	Source statement	
	1420	32 7E34	4079	STA CRCCN0	;Here
	1423	C9	4080	RET	;Leave
	1424	21 7E34	4081	1\$: LXI H,CRCCN0	;Point to this count
	1427	7E	4082	MOV A,M	;Get it
	1428	FE 01	4083	CPI 1	;Check for 0, 1, or 2
	142A	DA 1441	4084	JC 3\$	;Jump if 0
	142D	3A 7E32	4085	LDA XCRC	;Assume count=2
	1430	C2 1436	4086	JNZ 2\$	;Jump if 2
	1433	3A 7E33	4087	LDA XCRC+1	;Get the high byte instead
	1436	D3 40	4088	2\$: OUT TRDB	;Send it out
	1438	35	4089	DCR M	;Subtract one
	1439	C0	4090	RNZ	;Leave if not 0
	143A	21 0000	4091	LXI H,0	;Make a word of 0
	143D	22 7E32	4092	SHLD XCRC	;Zero out the Xmit CRC
	1440	C9	4093	RET	;Leave
	1441	21 7E76	4094	3\$: LXI H,MSGFLG	;See if the message type flag is on
	1444	7E	4095	MOV A,M	;Get it
	1445	A7	4096	ANA A	;Is it set?
	1446	CA 1458	4097	JZ 4\$	;Jump if clear
	1449	36 00	4098	MVI M,0	;Zero it
	144B	23	4099	INX H	;Point to the byte to be sent
	144C	7E	4100	MOV A,M	;Get it
	144D	D3 40	4101	OUT TRDB	;Send it
	144F	CD 13CE	4102	CALL XMTCRC	;Go CRC the character
	1452	3E 02	4103	MVI A,2	;Make a 2
	1454	32 7E35	4104	STA CRCCN1	;Set the the CRC count
	1457	C9	4105	RET	;Leave
	1458	21 7E63	4106	4\$: LXI H,CTL_COUNT	;Point to the CTL count.
	145B	7E	4107	MOV A,M	;Get the count.
	145C	A7	4108	ANA A	;Is it zero?
	145D	CA 146F	4109	JZ 5\$	;Jump if it is
	1460	35	4110	DCR M	;Reduce the count
	1461	2A 7E61	4111	LHLD CTL_PTR	;Get the address
	1464	7E	4112	MOV A,M	;Get the character
	1465	D3 40	4113	OUT TRDB	;Send it out
	1467	23	4114	INX H	;Bump the pointer
	1468	22 7E61	4115	SHLD CTL_PTR	;Save the new address
	146B	CD 13CE	4116	CALL XMTCRC	;Go CRC it
	146E	C9	4117	RET	;Leave
	146F	3A 7E27	4118	5\$: LDA OUTCNT	;Get the Output Buffer count
	1472	A7	4119	ANA A	;Is it zero?
	1473	CA 1488	4120	JZ 6\$	;Jump if it is
	1476	2A 7E28	4121	LHLD OUTBUF	;Get the address of what we are supposed to send
	1479	7E	4122	MOV A,M	;Get the character
	147A	D3 40	4123	OUT TRDB	;Send it out
	147C	23	4124	INX H	;Bump the pointer
	147D	22 7E28	4125	SHLD OUTBUF	;Save the new address
	1480	CD 13CE	4126	CALL XMTCRC	;Go CRC the character
	1483	21 7E27	4127	LXI H,OUTCNT	;Point to the count
	1486	35	4128	DCR M	;Subtract 1

Error Addr	Code	Seq	Source statement
1487	C9	4129	RET ;Leave
1488	21 7E35	4130	6\$: LXI H,CRCCN1 ;Point to this count
148B	7E	4131	MOV A,M ;Get it
148C	FE 01	4132	CPI 1 ;Look for 0, 1 or 2
148E	DA 149E	4133	JC 8\$;Jump if 0
1491	3A 7E32	4134	LDA XCRC ;Assume count is 2
1494	C2 149A	4135	JNZ 7\$;Jump if the count was 2
1497	3A 7E33	4136	LDA XCRC+1 ;Get the high byte
149A	D3 40	4137	7\$: OUT TRDB ;Send the character out
149C	35	4138	DCR M ;Reduce the count
149D	C9	4139	RET ;Leave
149E	3A 7E8F	4140	8\$: LDA LBSIP ;Get the Locked Buffer Send In Progress flag
14A1	32 7E90	4141	STA LBSENT ;Use it to set or clear the Locked Buffer Sent
		4142	;flag.
14A4	21 0000	4143	LXI H,0 ;Make a word of 0
14A7	22 7E32	4144	SHLD XCRC ;Zero out this CRC
14AA	AF	4145	XRA A ;Make a 0
14AB	32 7E07	4146	STA MSG_TO_SEND ;Clear this flag
14AE	21 7E3A	4147	LXI H,ENTER_TRANS ;Point to the Enter Trans Mode flag
14B1	BE	4148	CMP M ;See if it is 0 (A should contain 0 from the
		4149	;XRA A three lines up)
14B2	C8	4150	RZ ;Leave if it is
14B3	77	4151	MOV M,A ;Zero it now
14B4	32 7E06	4152	STA TRNFLAG ;and set Trans Mode
14B7	C3 10F8	4153	JMP TSQB ;Go see if we should send an ^.
		4154	;
		4155	;
		4156	;
		4157	;
		4158	;
		4159	;
		4160	;
14BA		4161	MSG_TR_DECODE:
14BA	21 7E2C	4162	LXI H,RD_LD_FLAG ;Point to the RD Load Flag.
14BD	BE	4163	CMP M ;See if it is zero.
14BE	C2 14DF	4164	JNZ DO_LD_COMMAND ;IF RD_LD_FLAG <> 0, THEN process the command
14C1	3A 7E00	4165	LDA REMRSR ;ELSE, get the Remote Rcv SR
14C4	A7	4166	ANA A ;Is there already a character in it?
14C5	C0	4167	RNZ ;IF CHARACTER PRESENT, THEN leave.
14C6	21 7E39	4168	LXI H,ASC_MSG_COUNT ;ELSE, is there an ASCII message to process?
14C9	BE	4169	CMP M ;Compare the zero in the AC with the count.
14CA	C2 1570	4170	JNZ GET_ASCII ;IF ASC_MSG_CNT <> 0, THEN go process it.
14CD	3A 7E46	4171	LDA MSG15_COUNT ;ELSE, how about Transparent Text?
14D0	A7	4172	ANA A ;Check it.
14D1	C2 0809	4173	JNZ STT_VECTOR ;IF MSG15_CNT <> 0, THEN go process it.
14D4	3A 40D0	4174	LDA SILO_ACTIVE ;ELSE, see if the silo has been emptied out
14D7	A7	4175	ANA A ;Has it?
14D8	C0	4176	RNZ ;Leave if it hasn't
14D9	32 7E1F	4177	STA MSG_TO_READ ;ZAP the Message To Be Read flag.
14DC	C3 16B5	4178	JMP UNLOCK_BUFFER ;Go unlock the input buffer.

Error Addr	Code	Seq	Source statement
		4179	
		4180	
14DF		4181	DO_LD_COMMAND:
14DF	21 7E4C	4182	LXI H, RD_LD_CCOUNT ;Point to the RD Load Packet Command Count.
14E2	7E	4183	MOV A, M ;Get it
14E3	A7	4184	ANA A ;Is it zero?
14E4	CA 14F3	4185	JZ 1\$;IF RD_LD_CCOUNT = 0, THEN jump.
14E7	35	4186	DCR M ;Subtract one.
14E8	2A 7E4A	4187	LHLD RD_LD_CPTR ;Get the Command Packet Pointer.
14EB	46	4188	MOV B, M ;Get the current byte
14EC	23	4189	INX H ;Bump the pointer.
14ED	22 7E4A	4190	SHLD RD_LD_CPTR ;Save the new address.
14F0	C3 1504	4191	JMP 2\$;Go send it and leave.
14F3	21 7E4F	4192	1\$: LXI H, RD_LD_DCOUNT ;Point to the RD Load Data Count.
14F6	7E	4193	MOV A, M ;Get it.
14F7	A7	4194	ANA A ;See if it is now zero.
14F8	CA 152D	4195	JZ FINISH_LD_CMND ;IF RD_LD_DCOUNT = 0, THEN go get the reply
		4196	;packet from the CPU.
14FB	35	4197	DCR M ;Reduce the count.
14FC	2A 7E4D	4198	LHLD RD_LD_DPTR ;Get the address of the byte to be sent.
14FF	46	4199	MOV B, M ;Get the byte.
1500	23	4200	INX H ;Bump the pointer.
1501	22 7E4D	4201	SHLD RD_LD_DPTR ;Save the new address.
1504	21 7E53	4202	2\$: LXI H, RD_LD_OE ;Point at the Odd/Even byte flag.
1507	7E	4203	MOV A, M ;Get it.
1508	2F	4204	CMA ;Flip the accumulator.
1509	E6 01	4205	ANI 1 ;Save only the LSB.
150B	77	4206	MOV M, A ;Store the result.
150C	78	4207	MOV A, B ;Get the byte to be sent.
150D	D3 CC	4208	OUT WRITE ;Put it in the WRITE register right now.
150F	0E 00	4209	MVI C, 0 ;Zap C without affecting the Condition Codes.
1511	CA 1519	4210	JZ 3\$;IF RD_LD_OE = 0, THEN jump to do odd byte.
1514	D3 AA	4211	OUT SETATN ;ELSE, this is the even word so Set Console
		4212	;Attention.
1516	C3 151B	4213	JMP 4\$;Jump over the next part.
1519	D3 AB	4214	3\$: OUT CLRATN ;Clear Console Attention.
151B	DB 82	4215	4\$: IN CPUACK ;Look for CPU Ack
151D	E6 01	4216	ANI 1 ;Save only the low bit.
151F	BE	4217	CMP M ;If we are sending the even byte, the odd/even
		4218	;flag will be a one and we are looking for
		4219	;CPUACK to be high. If it is the odd byte, the
		4220	;flag will be a zero and we looking for CPUACK
		4221	;to be low.
1520	C8	4222	RZ ;IF EQUAL, THEN leave.
1521	0D	4223	DCR C ;ELSE, decrement the count
1522	C2 151B	4224	JNZ 4\$;IF THE COUNT IS NOT 0, THEN loop.
1525		4225	SET_LD_ERROR:
1525	3E 01	4226	MVI A, 1 ;ELSE, if we fall through we have a Timeout.
1527	32 7E2D	4227	STA RD_LD_ERR ;Set this error flag.
152A	C3 156B	4228	JMP FINISH_LO ;Go clean things up and leave.

Error	Addr	Code	Seq	Source statement	
			4229		
			4230		
152D			4231	FINISH_LD_CMND:	
152D	0E 00		4232	MVI C,0	;Start the timeout count in C at zero.
152F	21 7E53		4233	LXI H, RD_LD_OE	;Point to the Odd/Even byte flag.
1532	7E		4234	MOV A,M	;Get it.
1533	2F		4235	CMA	;Flip it.
1534	E6 01		4236	ANI 1	;Save the LSB.
1536	77		4237	MOV M,A	;Store the result.
1537	DB 83		4238	1\$: IN CPATTN	;See if the uEngine is signaling us.
1539	E6 01		4239	ANI 1	;Save only the LSB.
153B	BE		4240	CMP M	;Check it out
153C	CA 1546		4241	JZ 2\$	;IF DONE, THEN leave the loop.
153F	0D		4242	DCR C	;Reduce the timeout count.
1540	C2 1537		4243	JNZ 1\$	;IF NOT TIMED OUT, THEN loop back.
1543	C3 1525		4244	JMP SET_LD_ERROR	;ELSE, go set the error flag.
1546	1F		4245	2\$: RAR	;Put the CPATTN bit in the carry.
1547	DB 80		4246	IN READ	;Get the byte.
1549	47		4247	MOV B,A	;Put it in B for now.
154A	DA 1552		4248	JC 3\$	;IF CPATTN = 1, THEN jump over the Clear Ack.
154D	D3 A9		4249	OUT CLRACK	;ELSE IF CPATTN = 0, clear Console Ack.
154F	C3 1554		4250	JMP 4\$	;Jump over the Set Acknowledge instruction.
1552	D3 A8		4251	3\$: OUT SETACK	;Since CPATTN is high, set Console Ack.
1554	21 7E52		4252	4\$: LXI H, RD_LD_ECOUNT	;Point to the End Packet count.
1557	7E		4253	MOV A,M	;Get it.
1558	35		4254	DCR M	;Subtract one from the count.
1559	CA 1564		4255	JZ 5\$	;IF ECOUNT = 0, THEN jump.
155C	FE 0A		4256	CPI 10	;Is this the first?
155E	C0		4257	RNZ	;Leave if not
155F	78		4258	MOV A,B	;Get the first byte the uEngine sent.
1560	32 7E54		4259	STA RD_LD_SUCCESS	;Save it here for later
1563	C9		4260	RET	;and leave.
1564	3A 7E54		4261	5\$: LDA RD_LD_SUCCESS	;Get the Success code byte.
1567	A7		4262	ANA A	;Was the transfer successful?
1568	C2 1525		4263	JNZ SET_LD_ERROR	;IF NOT SUCCESSFUL, THEN jump to set error flag.
			4264		;ELSE, fall through.
156B			4265	FINISH_LO:	
156B	AF		4266	XRA A	;Make a 0.
156C	32 7E2C		4267	STA RD_LD_FLAG	;Zap this flag.
156F	C9		4268	RET	;Leave.
			4269		
			4270		
			4271		
1570			4272	GET_ASCII:	
1570	35		4273	DCR M	;ELSE, subtract 1 from the count
1571	2A 7E37		4274	LHLD ASC_MSG_PTR	;Get the address into the Input Buffer
1574	7E		4275	MOV A,M	;Get the character
1575	23		4276	INX H	;Bump the pointer ahead
1576	22 7E37		4277	SHLD ASC_MSG_PTR	;and save it
1579	C3 1356		4278	JMP TRANS_REO	;Go share the code to write the data to REMRDB

Error	Addr	Code	Seq	Source statement
			4279	;
			4280	*****
			4281	
			4282	
			4283	
157C	21	7E02	4284	TRNSND: LXI H,REMXSR ;Point to the Remote Pseudo Xmit SR
157F	DB	41	4285	IN TRSR ;Get the Remote SR
1581	1F		4286	RAR ;See if Xmit Ready set
1582	D0		4287	RNC ;Leave with carry clear if not ready
1583	23		4288	INX H ;Point to the Remote pseudo DB
1584	7E		4289	MOV A,M ;Get the character
1585	D3	40	4290	OUT TRDB ;Send it the the USART
1587	2B		4291	DCX H ;Point to the pseudo SR
1588	36	01	4292	TRNSN1: MVI M,1 ;Set the ready bit
158A	AF		4293	XRA A ;Clear the carry to fake out the next
			4294	;conditional call
158B	C9		4295	RET ;Leave
			4296	
			4297	
			4298	*****
			4299	;
158C	2A	7E21	4300	PROSND: LHLD ACTCNT ;Get the count address
158F	7E		4301	MOV A,M ;Get it
1590	FE	50	4302	CPI 50H ;Check for Max
1592	CA	15C2	4303	JZ FULBUF ;IF FULL, THEN jump.
1595	34		4304	INR M ;ELSE, increment the count
1596	3A	7E8E	4305	LDA OUTLOC ;Get the Output Buffer Locked flag.
1599	A7		4306	ANA A ;Is it set?
159A	CA	15B2	4307	JZ 2\$;IF NOT LOCKED, THEN jump.
159D	3A	7E04	4308	LDA TALK_FLAG ;ELSE, check for Talk Mode.
15A0	1F		4309	RAR ;Look only at the Talk bit, not Talk Echo.
15A1	DA	15B2	4310	JC 2\$;IF TALK, don't try to send ^S.
15A4	7E		4311	MOV A,M ;ELSE, get the count again.
15A5	FE	38	4312	CPI 38H ;Are we exactly 24H from the end?
15A7	CA	15AF	4313	JZ 1\$;IF YES, THEN go try to send ^S.
15AA	FE	4D	4314	CPI 4DH ;ELSE, are we 4 or less from the end?
15AC	DA	15B2	4315	JC 2\$;IF NOT, THEN jump.
15AF	CD	111D	4316	1\$: CALL TESTSET_S ;ELSE, COUNT (=38H or >=4DH) so try to send ^S.
15B2	21	7E02	4317	2\$: LXI H,REMXSR ;Point to the Pseudo SR
15B5	36	01	4318	MVI M,1 ;Set the Xmit Ready
15B7	23		4319	INX H ;Point to the DB
15B8	7E		4320	MOV A,M ;Get the character
15B9	2A	7E23	4321	LHLD ACTBUF ;Get the address of the Output Buffer
15BC	77		4322	MOV M,A ;Store it
15BD	23		4323	INX H ;Bump the pointer
15BE	22	7E23	4324	SHLD ACTBUF ;Save the new version
15C1	C9		4325	RET ;
			4326	
			4327	*****
			4328	*****

Error	Addr	Code	Seq	Source statement	
			4329		
15C2	3A 7E8E		4330	FULBUF: LDA OUTLOC	;Get the Locked Buffer flag
15C5	A7		4331	ANA A	;Is it set?
15C6	C0		4332	RNZ	;Leave if it is.
15C7	21 7E8E		4333	FULBU0: LXI H,OUTLOC	;Point to the Locked Buffer flag
15CA	34		4334	INR M	;Set it
15CB	21 7E91		4335	LXI H,ACTFLG	;Point to the Active Buffer flag
15CE	7E		4336	MOV A,M	;Get it
15CF	32 7E8D		4337	STA LOBUF	;Indicate which output buffer is Locked
15D2	EE 01		4338	XRI 1	;Flip the Active Buffer flag
15D4	77		4339	MOV M,A	;Indicate which buffer is Active
15D5	11 7F5E		4340	LXI D,OCNT0	;We set up DE and HL here on the assumption that
15D8	21 7FAF		4341	LXI H,OCNT1	;it was a 1
15DB	CA 15DF		4342	JZ 1\$	;Jump if it was a 1
15DE	EB		4343	XCHG	;If it wasn't, eXCHanGe them
15DF	22 7E25		4344	1\$: SHLD LOCCNT	;Set up the Locked Buffer count address
			4345		; (The start of the Locked Buffer is implicitly
			4346		; the next location after the Locked Count
			4347		; address, so we don't need to save its address)
15E2	EB		4348	XCHG	;Put the correct pointer in HL
15E3	36 00		4349	MVI M,0	;Zero the count
15E5	22 7E21		4350	SHLD ACTCNT	;Store the address of the active count
15E8	23		4351	INX H	;Bump the pointer
15E9	22 7E23		4352	SHLD ACTBUF	;Store the address of the active buffer
15EC	C9		4353	RET	;Leave
			4354		
			4355		*****
			4356		*****
			4357		
			4358		
			4359		
			4360		*****
			4361		
			4362		
			4363		
			4364		
			4365		*****
			4366		
			4367	NACK	
			4368		
15ED			4369	NACK1:	
15ED	3A 7E1A		4370	NACK2: LDA DPFLAG	;Nack Reason 1 = Header CRC Error
			4371		;Nack Reason 2 = Message CRC Error
			4372		;Since the DP Flag will be 0 if we were doing a
			4373		;Header and 1 if we were doing a Data Message,
			4374		;we can use the incremented value for the Nack
			4375		;Reason
15F0	3C		4376	INR A	;Bump it
15F1	4F		4377	MOV C,A	;Put it in C
15F2	C3 1606		4378	JMP SNDNAK	;Go share the Send Nack code

Error	Addr	Code	Seq	Source statement
			4379	
15F5	0E 04		4380	NACK4: MVI C,4 ;Nack Reason 4 = Input Buffer Unavailable
15F7	C3 1606		4381	JMP SNDNAK ;Go share the Send Nack code
			4382	
15FA	0E 05		4383	NACK5: MVI C,5 ;Nack Reason 5 = Serial Line Error
15FC	C3 1606		4384	JMP SNDNAK ;Go share the Send Nack code
			4385	
15FF	0E 06		4386	NACK6: MVI C,6 ;Nack Reason 6 = Message Too Long
1601	C3 1606		4387	JMP SNDNAK ;
			4388	
1604	0E 07		4389	NACK7: MVI C,7 ;Nack Reason 7 = Header Format Error
1606	06 02		4390	SNDNAK: MVI B,2 ;Set up NACK code in B
1608	C5		4391	SNDNA0: PUSH B ;Save BC
1609	78		4392	MOV A,B ;Put the LAST_TYPE_SENT code in A
160A	CD 081B		4393	CALL BP_VECTOR ;Go prep things
160D	C1		4394	POP B ;Get BC back
160E	21 7EA7		4395	LXI H,ENQHED ;Point to this header
1611	22 7E95		4396	SHLD HDADR ;Set up this address
1614	36 05		4397	MVI M,5 ;Make the sure the first byte = ENQ
1616	23		4398	INX H ;Point to the second byte
1617	70		4399	MOV M,B ;Send out the CTL TYPE
1618	23		4400	INX H ;Point to the third byte
1619	71		4401	MOV M,C ;Send out the CTL REASON
161A	23		4402	INX H ;Point to the fourth byte
161B	3A 7E2A		4403	LDA LAST_GOOD_MSG ;Get the Last Good Data Message #
161E	77		4404	MOV M,A ;Set up the LAST MSG #
161F	23		4405	INX H ;Point to the fifth byte
1620	36 00		4406	MVI M,0 ;Zero the CURR MSG #
1622	23		4407	INX H ;Point to the sixth byte
1623	36 01		4408	MVI M,1 ;Make the ADDR a 1
1625	AF		4409	XRA A ;Make a 0
1626	32 7E30		4410	STA RCRC ;Clear the two Receive CRC bytes
1629	32 7E31		4411	STA RCRC+1 ;
162C	32 7E20		4412	STA SOHFLG ;Zero this flag
162F	32 7E1C		4413	STA HEADER ;Clear the Header flag
1632	32 7E1A		4414	STA DPFLAG ;and the Data Portion flag
1635	32 7E1D		4415	STA MESSAG ;And message flag
1638	3C		4416	INR A ;Make a 1
1639	32 7E98		4417	STA SOHENQ ;and set this one
163C	21 0008		4418	LXI H,8 ;Set up this count
163F	22 7E16		4419	SHLD ICNT ;
1642	21 7E68		4420	LXI H,HBUFFER ;Set up this address
1645	22 7E18		4421	SHLD IADDR ;
1648	79		4422	MOV A,C ;See if we are doing a NACK4
1649	FE 04		4423	CPI 4 ;Is it?
164B	C8		4424	RZ ;IF NACK4, THEN leave.
164C	3A 7E1F		4425	LDA MSG_TO_READ ;Get the Message to Read flag.
164F	A7		4426	ANA A ;Is it set?
1650	C0		4427	RNZ ;IF FLAG=1, THEN leave.
1651	C3 16B5		4428	JMP UNLOCK_BUFFER ;ELSE, unlock the Input Buffer.

Error	Addr	Code	Seq	Source statement
			4429	;
			4430	;
			4431	;
			4432	;
			4433	;
			4434	;
			4435	;
			4436	;
1654			4437	RD_COMMAND:
1654	1674		4438	DW RESERVED ;MSG TYPE 0 = Reserved
1656	167C		4439	DW INIT_PROTOCOL ;MSG TYPE 1 = Init Protocol
1658	1680		4440	DW EXIT_PROTOCOL ;MSG TYPE 2 = Exit Protocol
165A	1674		4441	DW RESERVED ;MSG TYPE 3 = Reserved
165C	1674		4442	DW RESERVED ;MSG TYPE 4 = Reserved
165E	0800		4443	DW RD_LD_VECTOR ;MSG TYPE 5 = LD Command
1660	1674		4444	DW RESERVED ;MSG TYPE 6 = Reserved
1662	1674		4445	DW RESERVED ;MSG TYPE 7 = Reserved
1664	1674		4446	DW RESERVED ;MSG TYPE 8 = Reserved
1666	1674		4447	DW RESERVED ;MSG TYPE 9 = Reserved
1668	1674		4448	DW RESERVED ;MSG TYPE 10 = Reserved
166A	1674		4449	DW RESERVED ;MSG TYPE 11 = Reserved
166C	1685		4450	DW NULL_MESSAGE ;MSG TYPE 12 = Null Message
166E	168E		4451	DW ASCII_MESSAGE ;MSG TYPE 13 = ASCII Message
1670	0803		4452	DW DCD_VECTOR ;MSG TYPE 14 = Diana Control Data Message
1672	0806		4453	DW TEXT_VECTOR ;MSG TYPE 15 = Transparent Text Message
			4454	;
			4455	;
			4456	;
			4457	;
			4458	;
			4459	RESERVED Message Types 0,3,4,6,7,8,9,10,11,16 and up
			4460	;
1674			4461	RESERVED:
1674	3E 01		4462	MVI A,1 ;Make a 1
1676	32 7E3B		4463	STA RDSPER ;Set the RD Service Protocol Error flag
1679	C3 16B5		4464	JMP UNLOCK_BUFFER ;Go unlock the Input Buffer if necessary.
			4465	;
			4466	;
			4467	;
			4468	;
			4469	;
			4470	;
			4471	INIT_PROTOCOL Message Type 1
			4472	;
167C			4473	INIT_PROTOCOL:
167C	CD 1682		4474	CALL IP_SUB ;Go Zap some things
167F	C3 10F8		4475	JMP TSQB ;Go see about sending ^Q.
1682	AF		4476	IP_SUB: XRA A ;Make a 0
1683	32 7E36		4477	STA NEEDI ;Clear the Need Init Protocol Message flag
1686	32 7E2A		4478	STA LAST_GOOD_MSG ;Clear the Last Good Data Message received

Error	Addr	Code	Seq	Source statement	
	1689	32 7E1F	4479	STA MSG_TO_READ	;and Message To Be Read flags
	168C	32 7E91	4480	STA ACTFLG	;Make buffer 0 the Active one
	168F	32 7E8F	4481	STA LBSIP	;Clear the Locked Buffer Send In Progress flag
	1692	32 7E90	4482	STA LBSENT	;and the Locked Buffer Sent flag
	1695	32 7E8E	4483	STA OUTLOC	;and the Locked Buffer flag
	1698	32 7E1B	4484	STA INLOCK	;Unlock the Input buffer
	169B	32 7E46	4485	STA MSG15_COUNT	;Zap this count
	169E	32 7E49	4486	STA CLMSG_COUNT	;and this one as well.
	16A1	32 7E2D	4487	STA RD_LD_ERR	;Clear this error flag.
	16A4	21 7F5E	4488	LXI H,OCNT0	;Point to the Output Buffer 0 Count
	16A7	77	4489	MOV M,A	;Zero the count
	16A8	22 7E21	4490	SHLD ACTCNT	;Set up this pointer
	16AB	23	4491	INX H	;Point to the beginning of the Buffer
	16AC	22 7E23	4492	SHLD ACTBUF	;and store the address here
	16AF	C9	4493	RET	;
			4494	:	
			4495	;	
			4496	:	
			4497	:	
			4498	;	
			4499	:	
			4500	;	
			4501	EXIT_PROTOCOL	Message Type 2
			4502	EXIT_PROTOCOL:	
16B0			4503	MVI A,1	;Make a 1
16B0	3E 01		4504	STA ENTER_TRANS	;Set the Enter Trans Mode flag
16B2	32 7E3A		4505		;and fall through to see about unlocking the
			4506		Input Buffer.
			4507	:	
			4508	:	
			4509	:	
			4510	:	
			4511	UNLOCK_BUFFER	
			4512	NULL_MESSAGE	Message Type 12
			4513	:	
16B5			4514	UNLOCK_BUFFER:	
16B5			4515	NULL_MESSAGE:	
16B5	3A 7E1F		4516	LDA MSG_TO_READ	;Do we still have a message being read?
16B8	A7		4517	ANA A	;Check the flag.
16B9	C0		4518	RNZ	;IF MSG_TO_READ = 1, then leave.
16BA	32 7E1B		4519	STA INLOCK	;ELSE, unlock the input buffer
16BD	C9		4520	RET	;Leave
			4521	:	
			4522	;	
			4523	:	
			4524	:	
			4525	;	
			4526	:	
			4527	;	
			4528	ASCII_MESSAGE	Message Type 13

Error Addr Code Seq Source statement

```
16BE      4529 ASCII_MESSAGE:
16BE 3A 7E40 4530 LDA IBUF_CNT_COPY ;Get this count
16C1 3D      4531 DCR A ;Subtract 1
16C2 CA 16B5 4532 JZ UNLOCK_BUFFER ;Go clean up because we are done
16C5 32 7E39 4533 STA ASC_MSG_COUNT ;Store the count here
16C8 2A 7E42 4534 LHLD IBUF_ADR_COPY ;Get the address of the buffer
16CB 23      4535 INX H ;Bump it ahead 1
16CC 22 7E37 4536 SHLD ASC_MSG_PTR ;and save it here
16CF      4537 ASCII_ME0:
16CF 3E 01    4538 MVI A,1 ;Make a 1
16D1 32 7E1F 4539 STA MSG_TO_READ ;Set the Message To Read flag
16D4 C9      4540 RET ;Leave
          4541 ;
          4542 ;*****
          4543
          4544
16D5      4545 CONLOST_MSG:
16D5 1E 0D 0A 4546 DB 30,CR,LF
16D8 3F 34 30 4547 DB '?40 (RD) CONNECTION LOST'
```

```
16DE 52 44 29
16E1 20 20 43
16E4 4F 4E 4E
16E7 45 43 54
16EA 49 4F 4E
16ED 20 4C 4F
16F0 53 54
16F2 0D 0A 4548 DB CR,LF
          4549
          4550
          4551 ORG 17FEH ;
          4552
17FE      4553 MSG17_MESSAGE:
17FE 01      4554 DB 1 ;Value of Message Type 17 Message.
          4555
          4556
          4557
          4558 ORG 17FFH ;
          4559
17FF 00      4560 HISUM: DB 0 ;
          4561
          4562
          4563
```

**
**
**
**
**
**
**
**

```
*****
;
; REVISION HISTORY
;
; 12.0 07-NOV-1986 Molly Bacon and Jim Sadin
; Fixed problem with unstable AC/DC LO with loaded UB.
; NOTE: Due to the unavailability of the original tools used to produce
; this file, this fix was hand edited into the hex code of the
; blasted roms. Therefore, parts of this file were also hand edited and not
; produced with an assembler. These lines are marked with ** in the first
; two columns. This file does represent exactly what is in the boot roms
; with part numbers as follows: 154F2 155F2 097F2 098F2 081F2 082F2
```

**

4568
4569
4570:
:
:11.10 30-SEPT-1982 vijay lathia(suggested by john middleton)
Following changes were made for future use when battrey back up

Error	Addr	Code	Seq	Source statement
			4571	;
			4572	;
			4573	;
			4574	;
			4575	;
			4576	;
			4577	;
			4578	;
			4579	;
			4580	;
			4581	;
			4582	;
			4583	;
			4584	;
			4585	;
			4586	;
			4587	;
			4588	;
			4589	;
			4590	;
			4591	;
			4592	;
			4593	;
			4594	;
			4595	;
			4596	;
			4597	;
			4598	;
			4599	;
			4600	;
			4601	;
			4602	;
			4603	;
			4604	;
			4605	;
			4606	;
			4607	;
			4608	;
			4609	;
			4610	;
			4611	;
			4612	;
			4613	;
			4614	;
			4615	;
			4616	;
			4617	;
			4618	;
			4619	;
			4620	;

unit will be installed. If changes were not made then interval timer will be initialized which should not be the case if BBU is present

#9

INIT:(+11)

11.09 09-SEPT-1982 Vijay Lathia(Suggested by Dave Maruska)

Following changes were made to take care of usart mode changing problem

#8

FIFTH_CHAR:(+2) & UBTST:(-3)

XRA A
OUT TCR
OUT TUCR
OUT TRCR

09.01 29-JULY-1982 John Middleton

Almost all of the changes between version 03.00 and version 09.01 have been RD related additions or fixes, mostly in the Protocol Mode area. One exception is that many of the variable, constant, and routine names have been changed to take advantage of the new assembler.

03.00 12-MAY-1982 John Middleton

#1

MODEM:

02.00 04-MAY-1982 John Middleton

#1

SOHDEC:

Here I made changes to prioritize the Nack 4 and Nack 6 error detection tests. Also I substituted locals for explicit names.

Error	Addr	Code	Seq	Source statement
			4621	
			4622	; 01.00 03-MAY-1982 John Middleton
			4623	;
			4624	; #1
			4625	;
			4626	; Changed the Self Test printout from CONSOLE to CONVnnn, where nnn is the
			4627	; version number of the ROM.
			4628	;
			4629	
			4630	
			4631	; 00.00 APRIL-1982 John Middleton
			4632	;
			4633	; #1
			4634	;
			4635	; Started version numbers back at 00.00
			4636	;
			4637	
			4638	
			4639	; 92.02 04-MAR-1982 John Middleton (Bug found by Vijay Lathia)
			4640	;
			4641	; #1
			4642	;
			4643	; UA7:(+6)
			4644	;
			4645	; Change JNZ to JNC because the preceeding instruction was changed from
			4646	; RAL to RLC.
			4647	
			4648	
			4649	; 85.01 30-NOV-1981 John Middleton
			4650	;
			4651	; #1
			4652	;
			4653	; SNDMS6:(-1)
			4654	;
			4655	; Don't want to fall through after decrementing the count.
			4656	; It would be faster than waiting for the next pass to find out
			4657	; the count = 0, but the Accumulator would have to be cleared
			4658	; for the ENTER_TRANS flag test to work.
			4659	
			4660	; 85.00 20-NOV-1981 John Middleton
			4661	;
			4662	; #1
			4663	;
			4664	; SNDMSA:(-3)
			4665	;
			4666	; Forgot to CRC the Message Type byte. Added call to XMTCRC
			4667	; routine.
			4668	;
			4669	;
			4670	; #2

* * Symbol Table * *

A	Reserved	ACLCHK	04CE	ACLO	4120	ACTBUF	7E23	ACTCNT	7E21
ACTFLG	7E91	APTFLG	0004	APTLOD	40D6	ASCII *	16CF	ASCII *	16BE
ASC_MS *	7E39	ASC_MS *	7E37	B	Reserved	BEGIN *	0000	BLDBLK *	7E9A
BOOTEN	0007	BOOTF	0001	BOOTPK	06B8	BP_VEC *	081B	BREAK	000D
BRKMSG	06CC	BRKSUB	03E0	BUFADR	40CA	BUILD *	0E2E	BUILD *	0E67
BUILD *	0818	BYTSUM	4082	C	Reserved	CAR_RE *	000D	CHAR_S *	4B80
CHKSUM *	00DE	CHRBUF	7001	CHRCNT	7000	CLEAR *	03C3	CLMSG *	7E49
CLMSG *	7E47	CLRACK	00A9	CLRACL	0033	CLRATN *	00AB	CLRBLK *	003A
CLRBSY *	002E	CLRCLK	0020	CLRCCK	0024	CLRCCK *	0038	CLRDCL	0031
CLRHLT	00AF	CLRLIT	003C	CLRMCI	0029	CLRMCK *	002B	CLRPFI	00AD
CLRRD	003E	CLRSS	0022	CLRTIM	00A5	CLRTMR *	002C	CLRUPC	00A8
CLRUPK	0026	CLRWCK	0036	CMSGNM	7E6C	CNTADR	40C8	CNTLC	054C
CNTLC0	0554	CNTLP	05AC	CNTLPO	05AF	CNTRLC *	0003	CNTRLO	000F
CNTRLP	0010	CNTRLQ	0011	CNTRLS	0013	CODE_F *	408B	COLD	408A
CONERR	124D	CONLOS *	16D5	CONTLC	06C6	CONTRP *	06C9	COPY	00F9
COPY1	00FB	CPATTN	0083	CPFLAG	40F3	CPUACK *	0082	CR	000D
CRCCN0	7E34	CRCCN1	7E35	CRCSUB	13D7	CSR07 *	0085	CSR15	0086
CSR23	0087	CTLFLG	7E97	CTLPRP	03C7	CTLREA *	7E6A	CTLTYP *	7E69
CTL_BI *	0D4D	CTL_BU *	7E64	CTL_CH *	40FE	CTL_CO *	7E63	CTL_C *	7E70
CTL_PT *	7E61	CVECTR	4C28	C_LETT *	0043	D	Reserved	DCD_CS *	0C31
DCD_CS *	0C4B	DCD_CS *	0C4C	DCD_FI *	0C5B	DCD_ME *	0C06	DCD_ME *	0BE1
DCD_SH *	0CD9	DCD_SS *	0C94	DCD-ST *	0CB1	DCD_VE *	0803	DCL0	0002
DELAY	1E62	DELETE *	40CE	DEVERR	06CE	DIRFLG *	409C	DISCAR *	7E05
DISCNT	40DA	DISMEM	00A7	DISPAR	00A1	DO_LD *	14DF	DPFLAG	7E1A
DPRINT	06A5	DROP_C *	117A	DROP_C *	115E	DRVMSG *	0700	DUMMY1	0052
E	Reserved	ENBMEM	00A6	ENBPAP	00A0	ENDPAK *	40AA	ENQDEC	1266
ENQHED	7EA7	ENTER *	7E3A	EPRIN1	0A1D	EPRIN2 *	0A27	EPRIN3	0A36
EPRINT	0A1B	EXIT_P *	16B0	FA_VEC *	1004	FBO_VE *	100D	FIFTH *	08A1
FINISH *	156B	FINISH *	152D	FINISH *	03B1	FIRST *	0850	FIXUP *	0691
FIX_AD *	101F	FLASH	112B	FLIP_O *	051D	FORCE *	059A	FOURTH *	089C
FULBU0	15C7	FULBUF	15C2	FU_VEC *	004F	GET_AP *	049F	GET_AS *	1570
GET_CH *	0427	GET_LO *	034B	GET_LO *	0346	GET_SI *	03F1	GOSV *	4103
GS_VEC *	0046	H	Reserved	HBUFFR	7E68	HCOUNT *	7E69	HCRC	7E6E
HDADR	7E95	HDCNT	7E94	HDECOD	125A	HEADER *	7E1C	HIGH_N *	0030
HISUM	17FF	HLBUFF	4088	HLTFLG	0002	HR_VEC *	1019	HR_VEC *	101C
IADDR	7E18	IBUF0	7EDB	IBUF1	7E60	IBUF_A *	7E42	IBUF_C *	7E40
ICNT	7E16	INDFLG	409D	INFIX	1298	INFIX0 *	129E	INFLAG	7E1E
INIT	0112	INIT0	0179	INIT1	0160	INIT2 *	017C	INIT3	0181
INIT4	01A5	INIT5	0190	INITH	0035	INITL *	0034	INITPK	06B6
INITX	0146	INIT_D *	0055	INIT_P *	167C	INIT_V *	0024	INIVEC	0040
INLOCK	7E1B	IP_SUB	1682	ITCSR	001D	ITDB *	001C	L	Reserved
LAST_G *	7E2A	LAST_T *	7E2B	LBSENT	7E90	LBSIP *	7E8F	LF	000A
LINE_F *	000A	LMSGNM	7E6B	LOBUF	7E8D	LOCAL *	7E71	LOCAL *	08B6
LOCCNT	7E25	LODADR	40A3	LODCNT	40A7	LODFLG *	409B	LOGCNT *	7E0F
LOGCON	7E0E	LOWSUM	07FF	LOW_NU *	0031	LR_MAS *	40BE	LR_MAS *	40BF
LR_MAS *	40BD	M	Reserved	MARCH	09A4	MARCH0 *	09AC	MARCH1	09B5
MARCH2	09BE	MARCH3	09C0	MARCH4	09D1	MARCH5 *	09EA	MARCH6	09F0
MARCH7	09FA	MARCH8	0A0B	MARCHX	09A8	MASK_G *	022E	MD_VEC *	0812
MESSAG	7E1D	MIDDLE *	0031	MIDSUM	0FFF	MIPFLG *	40D9	MISC2	0001
MOD0	1024	MOD0_V *	1007	MOD1	104A	MOD1A *	1052	MOD4	10B9

* * Symbol Table * *

MOD5 10C2
 MODEM4 10BC
 MODEM9 11C9
 MODVEC 0013
 MSG15 * 7E46
 MSGFLG 7E76
 MSG_TR * 14BA
 NACK5 15FA
 NEEDI 7E36
 NULL_M * 16B5
 OCNT1 7FAF
 OUTLOC 7E8E
 O_LETT * 004F
 POWER * 04EA
 PROSND 158C
 PWRVE1 004C
 RCRC 7E30
 RDCON 7E10
 RDSPER 7E3B
 RD_LD * 7E55
 RD_LD * 7E4F
 RD_LD * 7E2C
 RD_PAT * 1000
 REMDSR 10E0
 REMXSR 7E02
 RIPPLE 0988
 ROM_ER * 0340
 RST65 0034
 RSTN4 0020
 RUN_FL * 40D3
 SECOND * 0855
 SEND_C * 05D4
 SEND_T * 0B25
 SETBSY 002F
 SETHLT 00AE
 SETRD 003F
 SETUPK 0027
 SILO_A * 40D0
 SNDAOR 40A0
 SNDBAK 408D
 SOH_DE * 12BF
 SPEND 40D7
 ST_DON * 0A5D
 SYNTAX 0704
 TALK_C * 7E73
 TALK_S * 0D0F
 TEST_D * 10EE
 TICK1 1148
 TMRVEC 4C2F
 TRANS_ * 1356

MODEM0 0694
 MODEM5 10D1
 MODEM * 4099
 MRVE10 0824
 MSG15 * 7E44
 MSGTYP 7E77
 MYPC 40B9
 NACK6 15FF
 NOBOOT 06F2
 N_LETT * 004E
 OK15V 0007
 OVECTR 414E
 PAKCNT 40A8
 POWER * 04DF
 PROTO * 1375
 PWRVEC 001B
 RCVCRC 13D4
 RDCON1 7E11
 RDTEMP 7E12
 RD_LD * 7E4C
 RD_LD * 7E4D
 RD_LD * 7E53
 READ 0080
 REMOTE 0006
 RESERV * 1674
 RIPPLX 098C
 ROM_ID * 028F
 RST75 003C
 RSTN5 0028
 SAVESP 0104
 SECURE 0003
 SEND_C * 0B48
 SETACK 00A8
 SETCLK 0021
 SETLIT 003D
 SETSS 0023
 SETWCK 0037
 SILO_I * 40D1
 SNDCNT 40A2
 SOHENQ 7E98
 SOURCE * 4BC0
 STACK 4080
 ST_VEC * 0815
 S_FLAG 40CD
 TALK_F * 0D28
 TALK_V * 080F
 TEXT_M * 0CE2
 TIKTOK 7E0D
 TOCHK 01F9
 TRANS_ * 1349

MODEM1 105A
 MODEM6 119B
 MODEM * 0215
 MRVE11 0827
 MSG17 * 17FE
 MSG_DE * 0D88
 NACK1 15ED
 NACK7 1604
 NOCHK 40FF
 OBUF0 7F5F
 OK5V 0005
 OVERR1 * 7E67
 PARALL * 7E72
 PRIADR 40C1
 PSW Reserved
 QPEND 40D8
 RCVDON 0002
 RDD_VE * 100A
 RD_COM * 1654
 RD_LD * 0B70
 RD_LD * 7E52
 RD_LD * 7E54
 READJ2 0004
 REMRDB 7E01
 RETRY 408C
 RL_VEC * 080C
 ROM_ID * 0274
 RSTN1 0008
 RSTN6 0030
 SAVE_C * 40C0
 SECURE * 40F4
 SEND_I * 40F2
 SETACL 0032
 SETCSK 0025
 SETMCI 0028
 SETTIM 00A4
 SET_LD * 1525
 SILO_O * 40D2
 SNDMSG 10J1
 SOHFLG 7E20
 SP Reserved
 START 4100
 SUCCES 4116
 TALK 0CFB
 TALK_F * 0D29
 TEMP 40B4
 TEXT_V * 0806
 TIMER1 7E0A
 TOCNT 4083
 TRCR 0043

MODEM2 1076
 MODEM7 11A5
 MODEM * 068D
 MRVEC8 081E
 MSGADR 40B7
 MSG_TO * 7E1F
 NACK2 15ED
 NACK 0704
 NOTYPE 40D4
 OBUF1 7FB0
 OUTBUF 7E28
 O_FLAG 40BB
 PARERR 411D
 PRICNT 40C3
 PUBFLG 40B6
 RAM_LO * 0A6A
 RDAOK 1286
 RDNACK 12B8
 RD_DEC * 1314
 RD_LD * 7E4A
 RD_LD * 7E50
 RD_LD * 0800
 READ_S * 040F
 REMRSR 7E00
 RETRYI 40CC
 ROM_CH * 085B
 RPRMPT 0708
 RSTN2 0010
 RSTN7 0038
 SAVSP1 0108
 SELF_T * 082A
 SEND_L * 05B7
 SETATN 00AA
 SETCSR 0039
 SETMCK 002A
 SETTHR 002D
 SET_S * 03BB
 SIMMSK 000B
 SNDNA0 1608
 SOHFND 1368
 SPBUFF 4086
 STATIO * 7E6D
 SUMREG 0020
 TALKDB 7E14
 TALK_F * 7E04
 TESTSE * 1113
 THIRD * 085A
 TIMER2 7E0C
 TOCNT1 4084
 TRDB 0040

MODEM3 1083
 MODEM8 11B6
 MODM3A 10A3
 MRVEC9 0821
 MSGBUF 40F0
 MSG_TO * 7E07
 NACK4 15F5
 NACK_X * 070E
 NOT_X * 0338
 OCNT0 7F5E
 OUTCNT 7E27
 O_FLAG * 40BC
 PFFLAG 40B5
 PRINT * 0898
 PVECTR 4C2B
 RAM_VA * 4080
 RDAOK1 1289
 RDREP 1271
 RD_FLO * 11EA
 RD_LD * 0D7E
 RD_LD * 7E2D
 RD_LOA * 0BD9
 REDERR 06E1
 REMXDB 7E03
 RIPPL1 0993
 ROM_ER * 033D
 RST55 002C
 RSTN3 0018
 RSVEC 0049
 SCVECO 0043
 SEND_C * 05DA
 SEND_L * 05B9
 SETBLK 003B
 SETDCL 0030
 SETPFI 00AC
 SETUPC 00A9
 SET_S * 03B9
 SLWCLK 0006
 SNDNAK 1606
 SOHHED 7EA1
 SPECIA * 0B57
 STT_VE * 0809
 SWITCH * 4081
 TALKSR 7E13
 TALK_N * 0D2D
 TESTSE * 111D
 TICK 1140
 TIMSAV 7E08
 TOPREP 020B
 TRMODE 0042

* * Symbol Table * *

TRNFLG 7E06
TSQA 1105
TTCR 004B
TUCR 0047
TUINIT 0004
TU_RCV* 0AF2
UB1 0936
UB3A 097F
UNLOCK* 16B5
USART3* 08F9
V_LETT* 0056
WRITE 00CC
XVEC 000B

TRNSN1 1588
TSQA_V* 1016
TTCR_C* 0035
TUDB 0044
TUMODE 0046
TU_SEN* 00D1
UB1A 094B
UBTST 092E
UPC14 0084
VALIDI* 0713
WAIT 0512
XADRES 40C4
X_CMND* 02E9

TRNSND 157C
TSQB 10F8
TTDB 0048
TUERR 0086
TUSR 0045
TU_SEN* 0095
UB2 0950
UBX 0932
UPC14A 0000
VECTOR 06C4
WCSREG 0008
XCOUNT 40C6
YBUSRD 00EC

TRSR 0041
TSQ_VE* 1013
TTMODE 004A
TUERR2 0079
TU_RCV* 01DB
TU_XMI* 01C8
UB2A 0965
UB_VEC* 1001
USART1* 08B9
VNORML 00A3
WCS_LO* 40D5
XCRC 7E32

TRYTST 01B8
TSS_VE* 1010
TTSR 0049
TUERR3 0081
TU_RCV* 01D6
UA105 0984
UB3 096A
UNIT 4091
USART2* 08D9
VSHIFT 00A2
WORD_S* 409E
XMTCRC 13CE

Symbol Value References (# = Definition, \$ = Write, <BLANK> = Read)

A	Reserved	586\$	590\$	623\$	625\$	628\$	635\$	650\$	662	663	664\$	690	710	734
		735	737\$	742\$	745\$	753\$	755	757\$	762\$	765	786\$	788\$	794	801
		864	867\$	869\$	871\$	697\$	914	915	917	919	932\$	936	949\$	955\$
		957	960	966\$	971	974	976\$	994	1003	1019\$	1021	1023\$	1024	1025\$
		1038\$	1045\$	1048\$	1053	1058	1065\$	1070\$	1076\$	1082	1103	1128\$	1131\$	1132
		1139\$	1161	1164	1167	1169\$	1170\$	1173\$	1176\$	1179	1190\$	1194	1198	1206
		1215\$	1220\$	1233\$	1234\$	1240	1242\$	1245	1255	1258	1261	1296	1300	1301\$
		1303	1306\$	1316\$	1320\$	1326\$	1330	1347	1351	1352\$	1355	1358\$	1360	1370\$
		1373	1386\$	1400\$	1416\$	1430	1433	1436	1439\$	1443	1445\$	1447\$	1451	1459\$
		1475\$	1483\$	1487	1488	1490	1503	1509\$	1519\$	1529\$	1534	1537\$	1556\$	1578\$
		1588\$	1590	1591\$	1599\$	1603\$	1613\$	1619	1620\$	1626	1629	1633\$	1636\$	1639
		1680\$	1826\$	1830\$	1879\$	1881	1883\$	1886\$	1887	1896\$	1898	1900\$	1903\$	1904
		1915\$	1918\$	1920	1922\$	1925\$	1926	1949	1953\$	1983\$	1985	1997\$	2006\$	2023\$
		2032\$	2049\$	2058\$	2069\$	2071	2075	2085\$	2115\$	2117\$	2119\$	2125\$	2131\$	2133\$
		2135\$	2141\$	2147\$	2149\$	2151\$	2157\$	2164\$	2186\$	2189\$	2197	2218	2219	2221\$
		2227\$	2238\$	2241\$	2250\$	2253\$	2258\$	2268\$	2277\$	2285\$	2288\$	2297\$	2300\$	2329\$
		2334\$	2336	2337\$	2339\$	2342\$	2345\$	2347	2348\$	2349	2352\$	2381\$	2409\$	2413\$
		2425\$	2432\$	2436\$	2456\$	2459\$	2488	2502	2517	2520	2529\$	2532	2549	2561
		2563\$	2566\$	2569	2581	2589\$	2645\$	2658\$	2669	2676\$	2685\$	2689	2692\$	2704
		2705\$	2708	2709\$	2711\$	2715	2720	2723	2734\$	2742\$	2749\$	2755\$	2758	2765
		2766\$	2769\$	2776\$	2792\$	2798\$	2817\$	2819	2832\$	2843\$	2851	2854\$	2859\$	2890
		2892	2895\$	2897	2898\$	2902	2904	2905\$	2974	2975\$	2977	2979\$	2980	2984\$
		2987	2990	2993\$	2994	2999	3003\$	3006	3010	3011\$	3016	3021\$	3024	3026
		3030	3031	3034	3036	3041\$	3067\$	3070	3119	3128\$	3130\$	3205	3217\$	3224\$
		3226	3228\$	3240\$	3245	3255	3260\$	3263\$	3265\$	3305	3314	3319	3322\$	3327\$
		3328	3338\$	3339	3342	3351	3362\$	3370	3382	3387\$	3394\$	3396	3400\$	3404\$
		3408	3415\$	3427	3431\$	3435\$	3437	3440\$	3442	3443	3468	3473\$	3487\$	3488
		3499	3503\$	3506\$	3507	3546\$	3553	3564\$	3565	3567	3568	3570	3572	3584
		3590	3612	3615\$	3625	3627\$	3629\$	3630	3632\$	3636	3638\$	3644	3646\$	3650
		3672\$	3675\$	3677	3679	3680\$	3692	3753	3755\$	3758\$	3760	3761	3796\$	3821\$
		3822	3832	3838\$	3842\$	3847	3849\$	3854	3856\$	3859	3940	3966\$	3971	3979
		3980\$	3985	4023\$	4026	4032\$	4035	4037\$	4042	4067	4070\$	4078\$	4082\$	4095\$
		4096	4100\$	4103\$	4107\$	4108	4112\$	4119	4122\$	4131\$	4145	4151	4166	4172
		4175	4183\$	4184	4193\$	4194	4203\$	4206	4207\$	4226\$	4234\$	4237	4247	4253\$
		4258\$	4262	4266	4275\$	4289\$	4293	4301\$	4306	4311\$	4320\$	4322	4331	4336\$
		4339	4376\$	4377	4392\$	4404	4409	4416\$	4422\$	4426	4462\$	4476	4489	4503\$
		4517	4531\$	4538\$										
		526	1368\$											
		298\$	1374											
		361\$	4321	4324\$	4352\$	4492\$								
		360\$	2992	3321	4300	4350\$	4490\$							
		458\$	4335	4480\$										
		21\$	952	962	1658									
		270\$												
		4537\$												
		4451	4529\$											
		383\$	4168	4533\$										
		382\$	4274	4277\$	4536\$									
B	Reserved	663\$	687\$	693\$	781\$	864\$	867	871	1019	1021\$	1025	1032	1038	1073
		1131	1139	1165	1217\$	1218\$	1242	1259\$	1300\$	1316	1332\$	1351\$	1384\$	1385\$

ACLCHK 04CE
 ACLO 4120
 ACTBUF 7E23
 ACTCNT 7E21
 ACTFLG 7E91
 APTFLG 0004
 APTLOD 40D6
 ASCII_* 16CF
 ASCII_* 16BE
 ASC_MS* 7E39
 ASC_MS* 7E37

[illegible]

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)													
CLRTMR	002C	64#													
CLRUPC	00A8	152#													
CLRUPK	0026	55#													
CLRWCK	0036	79#													
CMSGNM	7E6C	436#	3861												
CNTADR	40C8	251#													
CNTLC	054C	1151	1459#												
CNTLCO	0554	1462#	1521												
CNTLP	05AC	1153	1518#												
CNTLP0	05AF	1135	1519#												
CNTRLC	0003	180#	1150	1318											
CNTRLO	000F	184#	1148												
CNTRLP	0010	181#	1152												
CNTRLQ	0011	183#	1146												
CNTRLS	0013	182#	1144												
CODE_F*	408B	203#	743\$	1464	2540										
COLD	408A	202#	767\$												
CONERR	124D	3475	3503#												
CONLOS*	16D5	3386	4545#												
CONTLC	06C6	1461	1712#												
CONTLP	06C9	1518	1713#												
COPY	00F9	687#													
COPY1	00FB	688#	782	2573											
CPATTN	0083	123#	4238												
CPFLAG	40F3	283#	1432	1460\$	1474	1520\$									
CPUACK	0082	121#	4215												
CR	000D	176#	1027	1041	1098	1509	1634	1723	1725	1727	1743	1746	1749	4546	
		4548													
CRCCN0	7E34	377#	3123\$	4079\$	4081										
CRCCN1	7E35	378#	3124\$	4104\$	4130										
CRCSUB	13D7	3930	3936#												
CSR07	0085	127#													
CSR15	0086	129#													
CSR23	0087	131#													
CTLFLG	7E97	463#	3008	3674\$											
CTLPRP	03C7	769	1206#	1471											
CTLREA	7E6A	433#													
CTLTYP	7E69	432#													
CTL_B1*	0D4D	2885#	3064												
CTL_BU*	7E64	415#	2908\$	2911\$											
CTL_CH*	40FE	287#	995\$	1049\$	1104\$	1136	1209\$								
CTL_CO*	7E63	414#	3059	3126\$	4106										
CTL_C*	7E70	442#	1321\$	3025	3032\$	3258\$									
CTL_PT*	7E61	413#	3063\$	4111	4115\$										
CVECTR	4C28	305#	1486												
C_LETT*	0043	1859	2931#												
D	Reserved	667\$	669\$	672	689	691\$	780\$	1873\$	1879	1881\$	1886	1896	1898\$	1903	
		1918	1920\$	1925	1985\$	1997	2003	2023	2029	2049	2055	2111\$	2115	2122	
		2131	2138	2147	2154	2181\$	2184	2190	2197\$	2226\$	2227	2242	2244\$	2248	
		2254	2256\$	2265\$	2267\$	2268	2280\$	2289	2291\$	2295	2301	2334	2336\$	2345	

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)												
		2346\$	2570\$	2576\$	2577\$	2580	2582\$	2644\$	2661\$	2664\$	2669\$	2671	2695	2701
		2731\$	2732	2780	2835	2847\$	2848	2852	2853\$	2997\$	3014	3015\$	3017	3575\$
		3629	3693\$	3695	3698\$	3939\$	3984	3985\$	4032	4035\$	4048	4340\$		
DCD_CS*	0C31	2650	2674#											
DCD_CS*	0C4B	2688#	2745											
DCD_CS*	0C4C	2678	2690#	2736										
DCD_FI*	0C5B	2698	2702#	2773										
DCD_ME*	0C06	2652#	2719	2726	2728									
DCD_ME*	0BE1	1776	2634#											
DCD_SH*	0CD9	2675	2733	2775#										
DCD_SS*	0C94	2649	2730#											
DCD_ST*	0CB1	2647	2748#											
DCD_VE*	0803	1775#	4452											
DCLO	0002	15#	1403	1412										
DELAY	1E62	173#												
DELETE*	40CE	257#	1507\$											
DEVERR	06CE	587	1723#											
DIRFLG	409C	213#												
DISCAR*	7E05	323#	1287	1609	2754\$	2779	2903	3257\$	3304					
DISCNT	40DA	277#	3216\$	3277										
DISMEM	00A7	143#												
DISPAR	00A1	134#												
DO_LD*	14DF	4164	4181#											
DPFLAG	7E1A	349#	3038\$	3208\$	3647\$	3846	4370	4414\$						
DPRINT	06A5	523	1676#											
DROP_C*	117A	3385	3392#											
DROP_C*	115E	3293	3296	3364	3379#									
DRVM5G	0700	808	1729#											
DUMMY1	0052	563#	563											
E	Reserved	662\$	669	670\$	1161\$	1169	1233	1255\$	1974\$	1984	2073\$	2117	2133	2149
		2185\$	2189	2228	2237\$	2241	2249\$	2253	2266\$	2269	2284\$	2288	2296\$	2300
		2346	2347\$	2679\$	2683\$	2684\$	2685	2732\$	2740\$	2741\$	2742	3546	3632	3638
		3692\$	3696\$	3937\$	3968	3978	3979\$	4037	4042\$	4050				
ENBMEM	00A6	142#												
ENBPAR	00A0	133#												
ENDPAK	40AA	226#	2396	2408										
ENQDEC	1266	3546#												
ENQHED	7EA7	475#	3552	4395										
ENTER*	7E3A	385#	4147	4504\$										
EPRIN1	0A1D	2326#	2328	2350										
EPRIN2	0A27	2331#	2333	2344										
EPRIN3	0A36	2338	2340#											
EPRINT	0A1B	2191	2243	2255	2290	2325#								
EXIT_P*	16B0	4440	4502#											
FA_VEC*	1004	1654	3173#											
FB0_VE*	100D	2995	3182#											
FIFTH*	08A1	1945	1947#											
FINISH*	156B	4228	4265#											
FINISH*	152D	4195	4231#											
FINISH*	03B1	1175#	1193	1428	1463									

Symbol	Value	References	(# = Definition, \$ = Write, <BLANK> = Read)
FIRST_*	0850	1848	1850#
FIXUP	0691	562	1654#
FIX_AD*	101F	3174	3200# 3230 3242 3247 3271 3279 3406 3410 3418
FLASH	112B	3236	3349#
FLIP_0*	051D	1149	1427#
FORCE	059A	1473	1503#
FOURTH*	089C	1941	1943#
FULBU0	15C7	3183	4333#
FULBUF	15C2	4303	4330#
FU_VEC*	004F	561#	3445
GET_AP*	049F	1274	1335#
GET_AS*	1570	4170	4272#
GET_CH*	0427	510	1250 1271#
GET_LO*	034B	1124#	1357 1361
GET_LO*	0346	1121#	1311 1314 1333
GET_SI*	03F1	558	1012 1050 1062 1079 1230#
GOSV	4103	293#	1085 1087
GS_VEC*	0046	557#	2812
H	Reserved	567\$	570\$ 576\$ 587\$ 591\$ 594\$ 627\$ 631\$ 637\$ 639\$ 641\$ 644\$ 647
		649\$	652\$ 674\$ 692\$ 713\$ 733\$ 736\$ 737 740\$ 779\$ 790\$ 806\$ 808\$
		813\$	821\$ 893\$ 897 900\$ 913\$ 916\$ 918\$ 931\$ 933\$ 938\$ 945\$ 997\$
		999\$	1001\$ 1022\$ 1033\$ 1044\$ 1056\$ 1060\$ 1065 1074\$ 1076 1083\$ 1088\$ 1101\$
		1125\$	1134\$ 1158\$ 1162\$ 1237\$ 1243\$ 1256\$ 1277\$ 1279\$ 1287\$ 1292\$ 1325\$ 1331\$
		1336\$	1342\$ 1435\$ 1438\$ 1442\$ 1461\$ 1465\$ 1467\$ 1474\$ 1482\$ 1484\$ 1489\$ 1518\$
		1530\$	1538\$ 1541\$ 1561\$ 1577\$ 1612\$ 1617\$ 1621\$ 1627\$ 1682\$ 1848\$ 1852\$ 1856\$
		1872\$	1873 1882\$ 1884 1899\$ 1901 1921\$ 1923 1941\$ 1945\$ 1975\$ 1983 2069
		2071\$	2108\$ 2178\$ 2182\$ 2214\$ 2217\$ 2220\$ 2221 2233\$ 2257\$ 2258 2265 2276\$
		2277	2363\$ 2366\$ 2379\$ 2393\$ 2396\$ 2408\$ 2412\$ 2417\$ 2420\$ 2424\$ 2426\$ 2428\$
		2431\$	2435\$ 2437\$ 2453\$ 2456 2463\$ 2465\$ 2469\$ 2473\$ 2495\$ 2511\$ 2531\$ 2541\$
		2545\$	2547\$ 2571\$ 2578\$ 2583\$ 2588\$ 2599\$ 2635\$ 2641\$ 2653\$ 2657\$ 2670\$ 2686\$
		2691\$	2700\$ 2743\$ 2764\$ 2779\$ 2796\$ 2814\$ 2816\$ 2825\$ 2829\$ 2831\$ 2837\$ 2842\$
		2844\$	2888\$ 2965\$ 2968\$ 2970\$ 2972\$ 2976\$ 2978\$ 2982\$ 3018\$ 3022\$ 3043\$ 3045\$
		3059\$	3061\$ 3201\$ 3215\$ 3238\$ 3267\$ 3269\$ 3326\$ 3330\$ 3335\$ 3337\$ 3362 3365\$
		3372\$	3374\$ 3380\$ 3386\$ 3389\$ 3393\$ 3402\$ 3415 3420\$ 3434\$ 3438\$ 3439\$ 3486\$
		3505\$	3563\$ 3569\$ 3571\$ 3576\$ 3581\$ 3582\$ 3614\$ 3616\$ 3619\$ 3641\$ 3643\$ 3684\$
		3694\$	3697\$ 3752\$ 3754\$ 3794\$ 3820\$ 3833\$ 3836\$ 3839 3841\$ 3843\$ 3929\$ 3932\$
		3938\$	4049\$ 4072\$ 4075\$ 4081\$ 4091\$ 4094\$ 4099\$ 4106\$ 4114\$ 4124\$ 4127\$ 4130\$
		4143\$	4147\$ 4162\$ 4168\$ 4182\$ 4189\$ 4192\$ 4200\$ 4202\$ 4233\$ 4252\$ 4276\$ 4284\$
		4288\$	4291\$ 4317\$ 4319\$ 4323\$ 4333\$ 4335\$ 4341\$ 4351\$ 4395\$ 4398\$ 4400\$ 4402\$
		4405\$	4407\$ 4418\$ 4420\$ 4488\$ 4491\$ 4535\$
HBUFFER	7E68	429#	3043 3269 3543 3576 4420
HCOUNT	7E69	431#	3541
HCRC	7E6E	438#	
HDADR	7E95	461#	3055\$ 4069 4073\$ 4396\$
HDCNT	7E94	460#	3131\$ 4066 4075
HDECOD	125A	3492	3541#
HEADER	7E1C	351#	3210\$ 3490 3585\$ 3850\$ 4413\$
HIGH_N*	0030	2216	2935#
HISUM	17FF	4560#	
HLBUFF	4088	200#	

Symbol	Value	References	(# = Definition, \$ = Write, <BLANK> = Read)
HLTFLG	0002	16#	
HR_VEC*	1019	3194#	3195
HR_VEC*	101C	3197#	3198
IADDR	7E18	347#	3044\$ 3270\$ 3577\$ 3831 3834\$ 4421\$
IBUF0	7EDB	477#	3643
IBUF1	7E60	409#	3641
IBUF_A*	7E42	390#	2574 2640 2643\$ 2656 2795 3578\$ 3671 4534
IBUF_C*	7E40	389#	2555 2635 2653 2791 3580\$ 3669 4530
ICNT	7E16	346#	3046\$ 3268\$ 3583\$ 3835 3837\$ 4419\$
INDFLG	409D	214#	1208\$
INFIX	1298	3551	3562 3575# 3592
INFIX0	129E	3577#	3649
INFLAG	7E1E	354#	3645\$ 3853
INIT	0112	513	733#
INIT0	0179	789#	
INIT1	0160	555	778# 1481
INIT2	017C	787	790# 815
INIT3	0181	792#	823
INIT4	01A5	807#	825
INIT5	0190	795	799#
INITH	0035	77#	1394\$ 1818\$
INITL	0034	76#	1399\$
INITPK	06B6	570	1693#
INITX	0146	760#	1375
INIT_D*	0055	565#	792
INIT_P*	167C	4439	4473#
INIT_V*	0024	512#	1835 2370
INIVEC	0040	555#	
INLOCK	7E1B	350#	3635 3857\$ 4484\$ 4519\$
IP_SUB	1682	3214	4474 4476#
ITCSR	001D	34#	746\$ 754\$
ITDB	001C	33#	747 750 756\$ 758\$
L	Reserved	742	898 1070 1163 1164\$ 1167\$ 1257 1258\$ 1261\$ 2000 2026 2052 2183
		2266	2459 3363 3416 3670 3838
LAST_G*	7E2A	368#	2973 3619 3862\$ 4403 4478\$
LAST_T*	7E2B	369#	3118\$ 3554
LBSENT	7E90	455#	3563 4141\$ 4482\$
LBSIP	7E8F	454#	3001\$ 4140 4481\$
LF	000A	177#	1636 4546 4548
LINE_F*	000A	178#	1851
LMSGNM	7E6B	435#	3614
LOBUF	7E8D	452#	4337\$
LOCAL_*	7E71	444#	2691 2707 2768\$ 2888 3211\$
LOCAL_*	08B6	1982#	2072 2074
LOCCNT	7E25	363#	3002 4344\$
LODADR	40A3	221#	2380\$ 2397\$ 2455 2464\$
LODCNT	40A7	222#	2391\$ 2406\$ 2453 2465
LODFLG	409B	212#	1482
LOGCNT	7E0F	335#	3264\$ 3486 3505
LOGCON	7E0E	334#	3398\$

[illegible]

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)
MODEM2	1076	3243#
MODEM3	1083	3248#
MODEM4	10BC	3272#
MODEM5	10D1	3280#
MODEM6	1198	3407#
MODEM7	11A5	3411#
MODEM8	11B6	3419#
MODEM9	11C9	3233 3427#
MODEM_*	4099	210# 1651 3202\$
MODEM_*	0215	502 929# 1275
MODEM_*	068D	930 1650#
MODM3A	10A3	3263#
MODVEC	0013	502#
MRVE10	0824	1806# 1806
MRVE11	0827	1808# 1808
MRVEC8	081E	1802# 1802
MRVEC9	0821	1804# 1804
MSG15_*	7E46	393# 1207\$ 1561 2495 2722 2794\$ 3313 3420 4171 4485\$
MSG15_*	7E44	392# 2490 2494\$ 2797\$
MSG17_*	17FE	2982 4553#
MSGADR	40B7	233# 593\$ 796 824
MSGBUF	40F0	279# 1462\$ 1472
MSGFLG	7E76	449# 3068\$ 3125\$ 4094
MSGTYP	7E77	450# 3066\$
MSG_DE*	0D88	1791 2950#
MSG_TO*	7E1F	356# 2565\$ 2799\$ 3456 4177\$ 4425 4479\$ 4516 4539\$
MSG_TO*	7E07	327# 3129\$ 3209\$ 3453 4146\$
MSG_TR*	14BA	3458 4161#
MYPC	40B9	234# 595\$
NACK1	15ED	3845 4369#
NACK2	15ED	4370#
NACK4	15F5	3640 4380#
NACK5	15FA	4383#
NACK6	15FF	3631 3634 4386#
NACK7	1604	3618 3626 3687 4389#
NACK_X*	0704	1740#
NACK_X*	070E	1083 1745#
NEEDI	7E36	380# 3613\$ 3628\$ 3678 3858 4477\$
NOBOOT	06F2	806 1727#
NOCHK	40FF	288# 1210\$ 1630
NOTYPE	40D4	266# 1211\$
NOT_X*	0338	1047 1097#
NULL_H*	16B5	4450 4515#
N_LETT*	004E	2110 2933#
OBUF0	7F5F	479#
OBUF1	7FB0	481#
OCNT0	7F5E	478# 4340 4488
OCNT1	7FAF	480# 4341
OK15V	0007	28# 1820
OK5V	0005	23#

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)									
OUTBUF	7E28	366#	3074\$	4121	4125\$						
OUTCNT	7E27	365#	3072\$	3127\$	4118	4127					
OUTLOC	7E8E	453#	2989	2998	3318	4305	4330	4333	4483\$		
OVECTR	414E	300#									
OVERRI*	7E67	422#	2670	3213\$							
O_FLAG	40BB	236#	789\$	996\$	1438	1450\$	1504\$	1574	2519		
O_FLAG*	40BC	237#	1449								
O_LETT*	004F	1944	2932#								
PAKCNT	40A8	224#	2382\$	2393							
PARALL*	7E72	445#	2703	2772\$	2886	3212\$					
PARERR	411D	297#	522								
PFFLAG	40B5	230#									
POWER*	04EA	560	1389#								
POWER*	04DF	507	761	1382#							
PRIADR	40C1	245#	1533\$	1536	1539\$						
PRICNT	40C3	246#	1532\$	1541							
PRINT*	0898	1917	1940#								
PROSND	158C	3465	4300#								
PROTO*	1375	3484	3819#								
PSW	Reserved	1029	1040\$	1369	1391	1392\$	3936	4051\$			
PUBFLG	40B6	231#	768\$								
PVECTR	4C2B	306#	1491	1494							
PWRVE1	004C	560#									
PWRVEC	001B	507#									
QPEND	40D8	273#	3335								
RAM_LO*	0A6A	1785	2376#								
RAM_VA*	4080	190#	1372	1465	1478	2545					
RCRC	7E30	374#	3039\$	3040\$	3586\$	3587\$	3795\$	3841	3932	4410\$	4411\$
RCVCRC	13D4	3830	3932#								
RCVDON	0002	185#	1297	1327	1348						
RDACK	1286	3548	3562#								
RDACK1	1289	3563#	3652								
RDCON	7E10	337#	766\$	931	969	1286	1606	2501	3261\$	3381	3397\$
RDCON1	7E11	338#	970\$								
RDD_VE*	100A	2951	3179#								
RDNACK	12B8	3550	3590#								
RDREP	1271	3551#									
RDSER	7E3B	387#	2986	3120\$	4463\$						
RDTEMP	7E12	340#	3472\$	3748	3823						
RD_COM*	1654	3694	4437#								
RD_DEC*	1314	3180	3668#								
RD_FLO*	11EA	3274	3452#								
RD_LD*	7E55	406#	2571	2578	2590\$						
RD_LD*	7E4C	399#	2567\$	4182							
RD_LD*	0B70	1773	2539#								
RD_LD*	7E4A	398#	2572\$	4187	4190\$						
RD_LD*	0D7E	2570	2920#								
RD_LD*	7E4F	401#	2560\$	2588	4192						
RD_LD*	7E4D	400#	2587\$	4198	4201\$						
RD_LD*	7E52	403#	2568\$	4252							

Symbol	Value	References	(# = Definition, \$ = Write, <BLANK> = Read)
RD_LD_*	7E50	402#	
RD_LD_*	7E2D	372#	2599 3027 3033\$ 3259\$ 4227\$ 4487\$
RD_LD_*	7E2C	371#	2564\$ 4162 4267\$
RD_LD_*	7E53	404#	2562\$ 4202 4233
RD_LD_*	7E54	405#	4259\$ 4261
RD_LD_*	0800	1772#	4443
RD_LOA*	0BD9	2553	2598#
RD_PAT*	1000	1661	3166#
READ	0080	119#	4246
READJ2	0004	20#	1833
READ_S*	040F	559	1231 1249#
REDERR	06E1	591	1725#
REMSR	10E0	3243	3248 3272 3280 3291# 3231 3291 3424
REMOTE	0006	26#	939 946 1337 1655
REMRDB	7E01	316#	3752
REMSR	7E00	315#	975\$ 1180\$ 1292 1325 4165
REMXDB	7E03	319#	
REMSR	7E02	318#	977\$ 1612 2825 2847 3459 4284 4317
RESERV*	1674	3689	443# 4441 4442 4444 4445 4446 4447 4448 4449 4461#
RETRY	408C	205#	790 821
RETRY1	40CC	254#	567 576
RIPPL1	0993	2184#	2188 2198 2354
RIPPLE	0988	2178#	
RIPPLX	098C	2178	2180#
RL_VEC*	080C	799	1784#
ROM_CH*	085B	1868#	
ROM_ER*	033D	1054	1100#
ROM_ER*	0340	1084	1089 1102#
ROM_ID*	028F	1011#	1013 1018 1028 1042
ROM_ID*	0274	495	992# 1099
RPRMPT	0708	997	1056 1743#
RST55	002C	517#	
RST65	0034	521#	
RST75	003C	525#	
RSTN1	0008	495#	
RSTN2	0010	500#	
RSTN3	0018	505#	
RSTN4	0020	510#	
RSTN5	0028	515#	
RSTN6	0030	519#	
RSTN7	0038	523#	
RSVEC	0049	559#	
RUN_FL*	40D3	264#	1125 1277 1342 2541 3341
SAVE\$P	0104	566	710# 2377
SAVE_C*	40C0	243#	1552\$ 1586 1597 1618 1625 1637\$
SAVSP1	0108	712#	
SCVECO	0043	556#	
SECOND*	0855	1852	1854#
SECURE	0003	18#	1276 1341 2550 2909
SECURE*	40F4	285#	

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)												
SELF_T*	082A	1794	1817#											
SEND_C*	05DA	556	1554#											
SEND_C*	05D4	505	1551#											
SEND_C*	0B48	2489	2506#											
SEND_I*	40F2	281#	1429	1437\$	1505\$	1557\$	1640\$							
SEND_L*	05B7	519	1528#											
SEND_L*	05B9	515	1531#											
SEND_T*	0B25	1782	2486#											
SETACK	00A8	145#	1870\$	1992\$	2018\$	2044\$	2113\$	2129\$	2145\$	2235\$	2246\$	2282\$	2293\$	4251\$
SETACL	0032	73#	1368\$	1383\$										
SETATN	00AA	155#	4211\$											
SETBLK	003B	86#	744\$											
SETBSY	002F	68#	1390\$	1819\$										
SETCLK	0021	47#												
SETCSK	0025	53#												
SETCSR	0039	83#												
SETDCL	0030	70#	1393\$											
SETHLT	00AE	161#												
SETLIT	003D	89#												
SETMCI	0028	58#												
SETMCK	002A	61#												
SETPFI	00AC	158#												
SETRD	003F	92#	3237\$	3357\$										
SETSS	0023	50#												
SETTIM	00A4	139#												
SETTMR	002D	65#												
SETUPC	00A9	153#												
SETUPK	0027	56#												
SETWCK	0037	80#												
SET_LD*	1525	4225#	4244	4263										
SET_S*	03BB	1191#	1199											
SET_S*	03B9	1145	1189#											
SILO_A*	40D0	260#	1154	1174\$	1212\$	1241\$	1251	4174						
SILO_I*	40D1	261#	1156	1172\$	1214\$	1237								
SILO_O*	40D2	262#	1158	1213\$	1236\$	1254								
SIMMSK	000B	186#	762											
SLWCLK	0006	25#	3366											
SNDADR	40A0	218#	630\$	634	638\$									
SNDCNT	40A2	219#	629\$	639	641									
SNDMSG	1401	3455	4063#											
SNDNA0	1608	3558	4391#											
SNDNAK	1606	3556	4378	4381	4384	4387	4390#							
SNDPAK	408D	207#	626\$	627	779	865	2421							
SOHENQ	7E98	465#	3042\$	3266\$	3820	4417\$								
SOHFLG	7E20	358#	3207\$	3476	3651\$	3744	3797\$	4412\$						
SOHFND	1368	3750	3794#											
SOHHED	7EA1	473#	2965	2997										
SOH_DE*	12BF	3545	3611#											
SOURCE*	4BC0	303#												
SP	Reserved	488\$	714	741	760\$	776	993\$							

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)											
SPBUFF	4086	199#	596	715\$	1469								
SPECIA*	0B57	2492	2508	2515#									
SPEND	40D7	272#	3326										
STACK	4080	172#	760	778	993	1467							
START	4100	292#	803	805	2379								
STATIO*	7E6D	437#											
STT_VE*	0809	1781#	3422	4173									
ST_DON*	0A5D	2279	2362#										
ST_VEC*	0815	490	1793#										
SUCCESS	4116	295#	592\$	711\$	793	800	2440\$						
SUMREG	0020	36#											
SWITCH*	4081	192#	938	950\$	961\$	1336							
SYNTAX	0704	1101	1741#										
S_FLAG	40CD	255#	1192\$	1506\$	1571	2516							
TALK	0CFB	1788	2809#										
TALKDB	7E14	343#	2814	2831									
TALKSR	7E13	342#	978\$	2809	2842								
TALK_C*	7E73	447#	2639\$	2713									
TALK_F*	0D28	2834#	2855										
TALK_F*	0D29	2836#	2861										
TALK_F*	7E04	321#	958\$	1014	1122	1279	1309	1566	2498	2638	2686	2716	2743
		2821	2891	3256\$	3310	3393	3428\$	4308					2764
TALK_N*	0D2D	2823	2841#										
TALK_S*	0D0F	2811	2820#										
TALK_V*	080F	1017	1569	1787#									
TEMP	40B4	228#											
TESTSE*	1113	3189	3307	3316	3320	3325#	3429						
TESTSE*	111D	2800	3186	3306	3334#	3391	4316						
TEST_D*	10EE	3303#	3384	3423									
TEXT_M*	0CE2	1779	2790#										
TEXT_V*	0806	1778#	4453										
THIRD_*	085A	1856	1858#										
TICK	1140	3246	3251	3254	3282	3361#	3500						
TICK1	1148	3349	3365#	3409	3417								
TIKTOK	7E0D	332#	3227\$	3365									
TIMER1	7E0A	330#	3239\$	3275	3278\$	3284\$	3361	3371	3373\$	3403\$	3414		
TIMER2	7E0C	331#	3229\$	3241\$	3244	3350	3374	3405\$	3407				
TIMSAV	7E08	329#	3276\$	3283									
TMRVEC	4C2F	308#	518										
TOCHK	01F9	845	873	893#									
TOCNT	4083	196#	893	913									
TOCNT1	4084	197#	896	901\$									
TOPREP	020B	572	840	856	913#								
TRANS_*	1356	3751#	4278										
TRANS_*	1349	3483	3743#										
TRCR	0043	97#	956\$	967\$	1359\$	1825	1952\$	1956\$	2078\$	2120\$	2136\$	2148\$	2153
		3401\$	3432\$	3504\$									3225\$
TRDB	0040	94#	1350	1598\$	2050\$	2054	3471	4071\$	4088\$	4101\$	4113\$	4123\$	4137\$
TRMODE	0042	96#	1829\$										4290\$
TRNFLG	7E06	325#	3206\$	3462	3481	3648\$	4152\$						

Symbol	Value	References (# = Definition, \$ = Write, <BLANK> = Read)													
TRNSN1	1588	4292#													
TRNSND	157C	3464	4284#												
TRSR	0041	95#	1346	1594	2046	2051	3234	3252	3294	3411	3467	4063	4285		
TRYTST	01B8	802	820#												
TSQA	1105	3192	3262	3318#											
TSQA_V*	1016	2503	2724	3191#											
TSQB	10F8	3310#	3573	4153	4475										
TSQ_VE*	1013	3188#													
TSS_VE*	1010	2727	3185#												
TTCR	004B	113#	1308\$	1824	1832\$	1950\$	1954\$	2076\$	2086\$	2116\$	2121	2134\$	2152\$	2166\$	
TTCR_C*	0035	187#	1306	1830	2164										
TTDB	0048	108#	1299	1587\$	1681\$	1998\$	2002	2330\$	2340\$	2530\$	2833\$	2860\$			
TTMODE	004A	110#	1828\$												
TTSR	0049	109#	1290	1295	1583	1677	1683	1994	1999	2326	2331	2522	2824	2856	
TUCR	0047	105#	1216\$	1221\$	1371\$	1823	1951\$	1955\$	2077\$	2118\$	2132\$	2137	2150\$		
TUDB	0044	100#	636\$	651\$	863	870\$	1222	2024\$	2028						
TUERR	0086	588	592#												
TUERR2	0079	579	586#	899											
TUERR3	0081	590#	2385	2389	2400	2404	2411	2415	2419	2423	2430	2434	2439	2471	
		2476													
TUINIT	0004	174#	1693	1693											
TUMODE	0046	102#	1827\$												
TUSR	0045	101#	842	860	2020	2025									
TU_RCV*	010B	573	858#	874											
TU_RCV*	01D6	500	855#												
TU_RCV*	0AF2	2392	2407	2451#	2467										
TU_SEN*	00D1	571	645#												
TU_SEN*	0095	620#	2378												
TU_XMI*	01C8	633	648	839#											
UA105	0984	2164#													
UB1	0936	2113#	2127												
UB1A	094B	2123	2125#												
UB2	0950	2129#	2143												
UB2A	0965	2139	2141#												
UB3	096A	2145#	2159												
UB3A	097F	2155	2157#												
UBTST	092E	2108#													
UBX	0932	2108	2110#												
UB_VEC*	1001	2558	2559	2601	2637	2655	2667	2672	2793	3169#					
UNIT	4091	208#	810	813	2416										
UNLOCK*	16B5	3170	4178	4428	4464	4514#	4532								
UPC14	0084	125#													
UPC14A	0000	10#													
USART1*	08B9	1991#	2008												
USART2*	08D9	2017#	2034												
USART3*	08F9	2043#	2060												
VALIDI*	0713	1088	1748#												
VECTOR	06C4	1705#													
VNORML	00A3	137#													
VSHIFT	00A2	136#													

[illegible]